HOMEWORK 5 REPORT – WATER QUALITY

Whether water is utilized for drinking, domestic use, food production, or recreational activities, it is crucial for the public's health to have access to safe, readily available water. Improved water supply, sanitation, and water resource management can significantly lower poverty while also boosting a country's economic growth. Therefore, predicting water quality based on its features with the help of certain ML tools can provide major insights.

AIM: To predict the Potability of water by utilizing its characteristics and identifying the model with the better accuracy by varying its parameters.

A. Data gathering and integration:

```
library(tidyverse)

library(dplyr)

library(rpart)

library(caret)

library(rattle)

library(varImp)

library(stats)

library(e1071)

library(astsa)

library(readxl)

library(factoextra)

library(ggplot2)

library(kknn)

library(cluster)

library(GGally)

library(pROC)
```

Loading the dataset and finding out its characteristics:

```
df_wp <- read.csv("C:/Users/nachi/Desktop/Fundamentals/HW_5/water_potability.
csv")
head(df_wp) #view the first 6 rows of the dataset
```

```
##          ph Hardness   Solids Chloramines  Sulfate Conductivity Organic_car
bon
## 1       NA 204.8905 20791.32    7.300212 368.5164     564.3087      10.379
783
## 2 3.716080 129.4229 18630.06    6.635246       NA     592.8854      15.180
013
## 3 8.099124 224.2363 19909.54    9.275884       NA     418.6062      16.868
637
## 4 8.316766 214.3734 22018.42    8.059332 356.8861     363.2665      18.436
524
## 5 9.092223 181.1015 17978.99    6.546600 310.1357     398.4108      11.558
279
## 6 5.584087 188.3133 28748.69    7.544869 326.6784     280.4679       8.399
735
##   Trihalomethanes Turbidity Potability
## 1        86.99097  2.963135          0
## 2        56.32908  4.500656          0
## 3        66.42009  3.055934          0
## 4       100.34167  4.628771          0
## 5        31.99799  4.075075          0
## 6        54.91786  2.559708          0
```

```
summary(df_wp) #summary
```

```
##        ph            Hardness          Solids         Chloramines
##  Min.   : 0.000   Min.   : 47.43   Min.   :  320.9   Min.   : 0.352
##  1st Qu.: 6.093   1st Qu.:176.85   1st Qu.:15666.7   1st Qu.: 6.127
##  Median : 7.037   Median :196.97   Median :20927.8   Median : 7.130
##  Mean   : 7.081   Mean   :196.37   Mean   :22014.1   Mean   : 7.122
##  3rd Qu.: 8.062   3rd Qu.:216.67   3rd Qu.:27332.8   3rd Qu.: 8.115
##  Max.   :14.000   Max.   :323.12   Max.   :61227.2   Max.   :13.127
##  NA's   :491
##     Sulfate        Conductivity    Organic_carbon  Trihalomethanes
##  Min.   :129.0   Min.   :181.5   Min.   : 2.20   Min.   :  0.738
##  1st Qu.:307.7   1st Qu.:365.7   1st Qu.:12.07   1st Qu.: 55.845
##  Median :333.1   Median :421.9   Median :14.22   Median : 66.622
##  Mean   :333.8   Mean   :426.2   Mean   :14.28   Mean   : 66.396
##  3rd Qu.:360.0   3rd Qu.:481.8   3rd Qu.:16.56   3rd Qu.: 77.337
##  Max.   :481.0   Max.   :753.3   Max.   :28.30   Max.   :124.000
##  NA's   :781                                     NA's   :162
##    Turbidity        Potability
##  Min.   :1.450   Min.   :0.0000
##  1st Qu.:3.440   1st Qu.:0.0000
##  Median :3.955   Median :0.0000
##  Mean   :3.967   Mean   :0.3901
##  3rd Qu.:4.500   3rd Qu.:1.0000
##  Max.   :6.739   Max.   :1.0000
##
```

```
str(df_wp)  #data types

## 'data.frame':    3276 obs. of  10 variables:
##  $ ph             : num  NA 3.72 8.1 8.32 9.09 ...
##  $ Hardness       : num  205 129 224 214 181 ...
##  $ Solids         : num  20791 18630 19910 22018 17979 ...
##  $ Chloramines    : num  7.3 6.64 9.28 8.06 6.55 ...
##  $ Sulfate        : num  369 NA NA 357 310 ...
##  $ Conductivity   : num  564 593 419 363 398 ...
##  $ Organic_carbon : num  10.4 15.2 16.9 18.4 11.6 ...
##  $ Trihalomethanes: num  87 56.3 66.4 100.3 32 ...
##  $ Turbidity      : num  2.96 4.5 3.06 4.63 4.08 ...
##  $ Potability     : int  0 0 0 0 0 0 0 0 0 0 ...

nrow(df_wp) #number of rows

## [1] 3276
```

__The Features within the dataset are:__

1.**ph:** pH of 1. water (0 to 14).

2.**Hardness:** Capacity of water to precipitate soap in mg/L.

3.**Solids:** Total dissolved solids in ppm.

4.**Chloramines:** Amount of Chloramines in ppm.

5.**Sulfate**: Amount of Sulfates dissolved in mg/L.

6.**Conductivity:** Electrical conductivity of water in μS/cm.

7.**Organic_carbon**: Amount of organic carbon in ppm.

8.**Trihalomethanes:** Amount of Trihalomethanes in μg/L.

9.**Turbidity:** Measure of light emiting property of water in NTU.

10.**Potability**: Indicates if water is safe for human consumption. Potable - 1 and Not potable - 0

As you cansee the dataset consists of substantial amount of missing values- 1.pH - NA's :491 2.Sulfate - NA's :781 3.Trihalomethanes - NA's :162 From further the rows with missing values will be removed to consider accurate analysis.

All the features are numerical except for- “Potability” which is of type, we will be converting this to character type and consider this variable as our **Target variable** to perform various prediction models and other clustering techniques.

0 - NonPotable

1 - Potable

B.DATA EXPLORATION:

```
# Categorizig column potability as “Potable” – 1 and “NonPotable” – 0
df_wp$Potability<-replace(df_wp$Potability,df_wp$Potability==0,”NonPotable”)
df_wp$Potability<-replace(df_wp$Potability,df_wp$Potability==1,”Potable”)
```

```
summary(df_wp) #summary
```

```
##        ph            Hardness          Solids          Chloramines
##  Min.   : 0.000   Min.   : 47.43   Min.   :  320.9   Min.   : 0.352
##  1st Qu.: 6.093   1st Qu.:176.85   1st Qu.:15666.7   1st Qu.: 6.127
##  Median : 7.037   Median :196.97   Median :20927.8   Median : 7.130
##  Mean   : 7.081   Mean   :196.37   Mean   :22014.1   Mean   : 7.122
##  3rd Qu.: 8.062   3rd Qu.:216.67   3rd Qu.:27332.8   3rd Qu.: 8.115
##  Max.   :14.000   Max.   :323.12   Max.   :61227.2   Max.   :13.127
##  NA’s   :491
##     Sulfate       Conductivity   Organic_carbon  Trihalomethanes
##  Min.   :129.0   Min.   :181.5   Min.   : 2.20   Min.   :  0.738
##  1st Qu.:307.7   1st Qu.:365.7   1st Qu.:12.07   1st Qu.: 55.845
##  Median :333.1   Median :421.9   Median :14.22   Median : 66.622
##  Mean   :333.8   Mean   :426.2   Mean   :14.28   Mean   : 66.396
##  3rd Qu.:360.0   3rd Qu.:481.8   3rd Qu.:16.56   3rd Qu.: 77.337
##  Max.   :481.0   Max.   :753.3   Max.   :28.30   Max.   :124.000
##  NA’s   :781                                     NA’s   :162
##    Turbidity      Potability
##  Min.   :1.450   Length:3276
##  1st Qu.:3.440   Class :character
##  Median :3.955   Mode  :character
##  Mean   :3.967
##  3rd Qu.:4.500
##  Max.   :6.739
##

str(df_wp)  #data types

## ‘data.frame’:    3276 obs. Of  10 variables:
##  $ ph             : num  NA 3.72 8.1 8.32 9.09 …
##  $ Hardness       : num  205 129 224 214 181 …
##  $ Solids         : num  20791 18630 19910 22018 17979 …
##  $ Chloramines    : num  7.3 6.64 9.28 8.06 6.55 …
##  $ Sulfate        : num  369 NA NA 357 310 …
##  $ Conductivity   : num  564 593 419 363 398 …
##  $ Organic_carbon : num  10.4 15.2 16.9 18.4 11.6 …
##  $ Trihalomethanes: num  87 56.3 66.4 100.3 32 …
##  $ Turbidity      : num  2.96 4.5 3.06 4.63 4.08 …
##  $ Potability     : chr  "NonPotable" "NonPotable" "NonPotable" "NonPotabl
e" …

nrow(df_wp)

## [1] 3276

df_wp %>% select(ph,Hardness,Solids,Chloramines,Sulfate,Conductivity,Organic_
carbon,Trihalomethanes,Turbidity) %>% ggpairs()
```

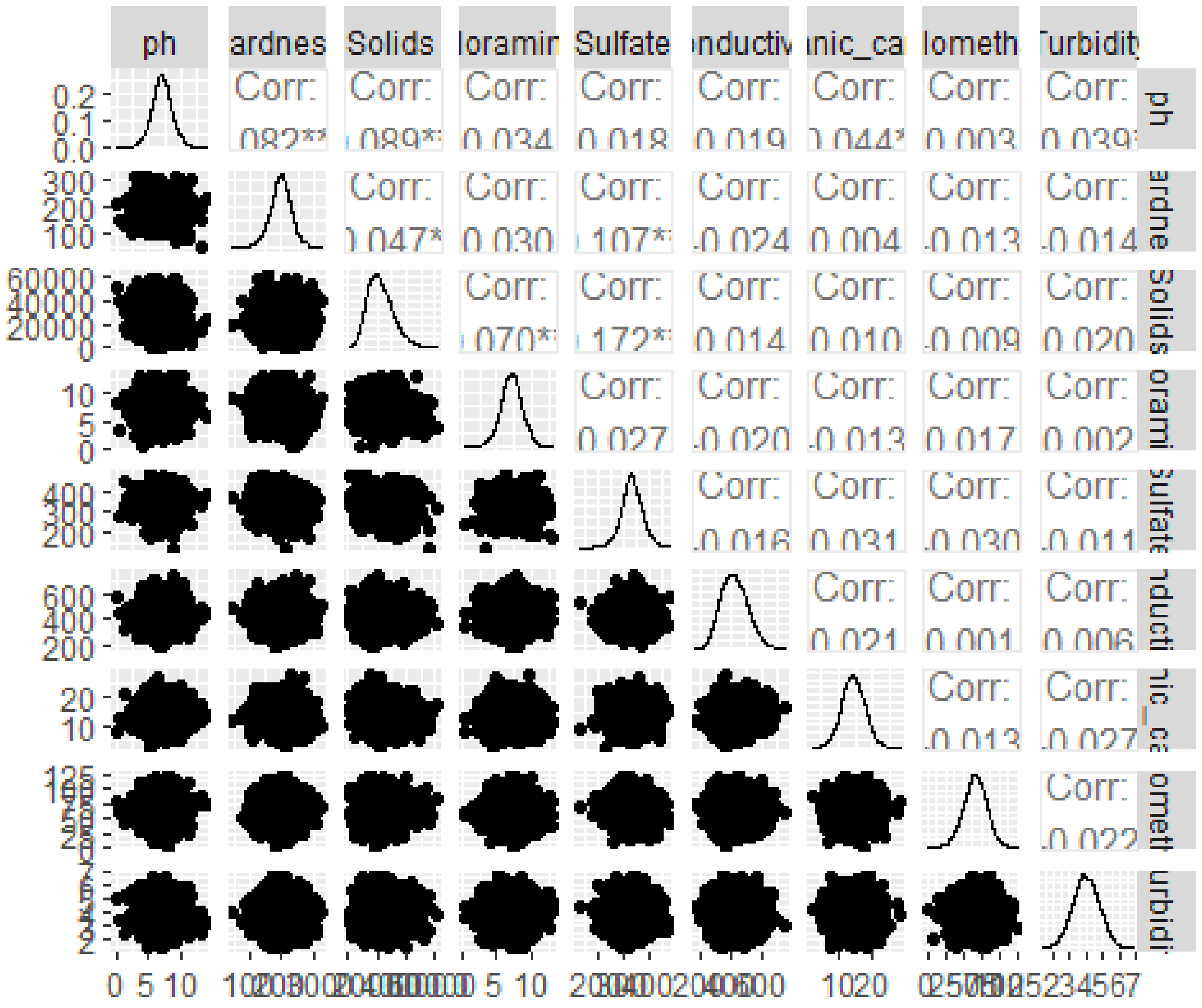


Figure 1. Correlation of features,

We can look at different correlations between them using Pearson's correlation; if one has a high correlation, we can remove the connected variable. Since the correlation value is below the threshold of 0.75, multicollinearity is likely not present currently.

```
ggcorr(df_wp)
```

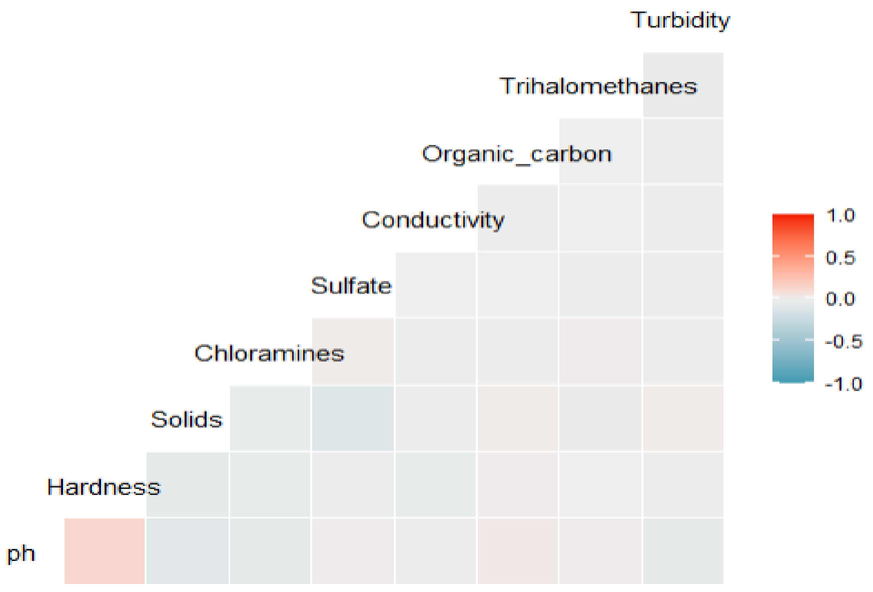


Figure 2 ggcorr plot to check correlation

C. DATA CLEANING:

STEP 1 REPLACE MISSING VALUES

```
summary(df_wp$ph)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   0.000   6.093   7.037   7.081   8.062  14.000     491

summary(df_wp$Sulfate)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   129.0   307.7   333.1   333.8   360.0   481.0     781

summary(df_wp$Trihalomethanes)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   0.738  55.845  66.622  66.396  77.337 124.000     162

df_wp$ph<- df_wp$ph %>%
    replace_na(median(df_wp$ph, na.rm = TRUE))
# Make sure the missing values are handled
summary(df_wp$ph)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   6.278   7.037   7.074   7.870  14.000

df_wp$Sulfate<- df_wp$Sulfate %>%
    replace_na(median(df_wp$Sulfate, na.rm = TRUE))
# Make sure the missing values are handled
summary(df_wp$Sulfate)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   129.0   317.1   333.1   333.6   350.4   481.0

df_wp$Trihalomethanes<- df_wp$Trihalomethanes %>%
    replace_na(median(df_wp$Trihalomethanes, na.rm = TRUE))
# Make sure the missing values are handled
summary(df_wp$Trihalomethanes)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.738  56.648  66.622  66.407  76.667 124.000
```

We can see that the difference between the mean and median values is minimal. So, we use the overall median of the feature to impute the values.

STEP 2 CHECKING FOR OUTLIERS AND REMOVING THEM

FURTHER DATA EXPLORATION HAS BEEN PERFORMED AS WELL (B.DATA EXPLORATION)

Plot of potability

```
ggplot(df_wp,aes(factor(Potability))) +
  geom_bar(alpha = 0.5,fill = 'darkslategray3') +
  theme_classic() + coord_flip()
```

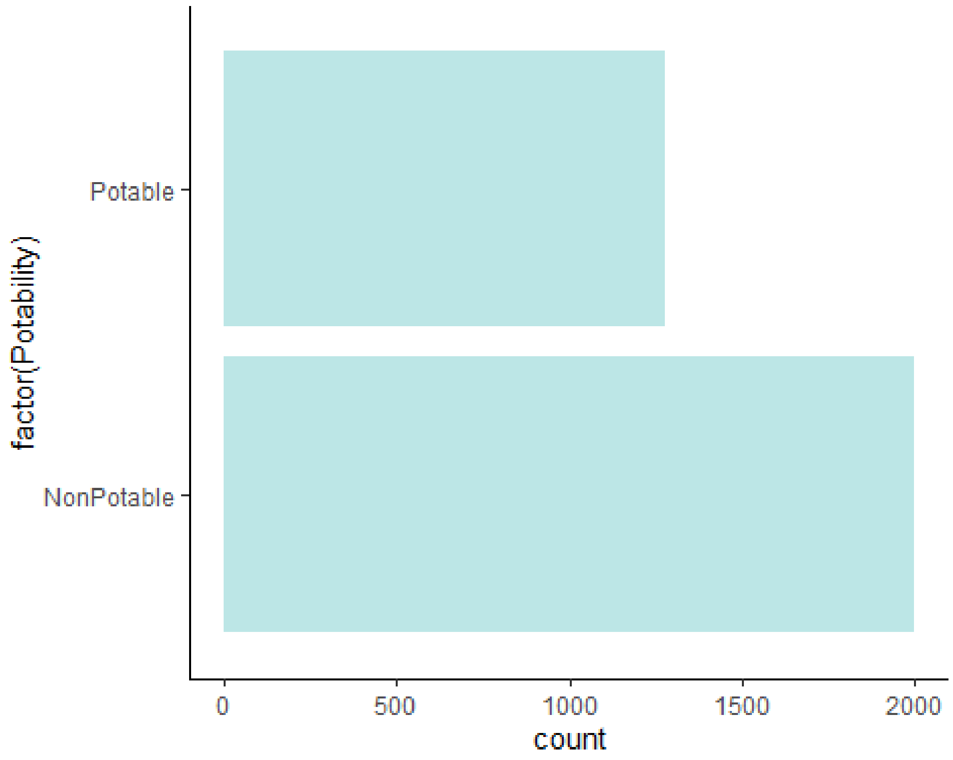


Figure 3 Potable & Non Potable Count

```
#There is a slight imbalance in the dataset
```

Lets take a look at each feature with respect to potability

Here we compare the before and removing outliers each feature, it is necessary to remove outliers to improve accuracy of a model.

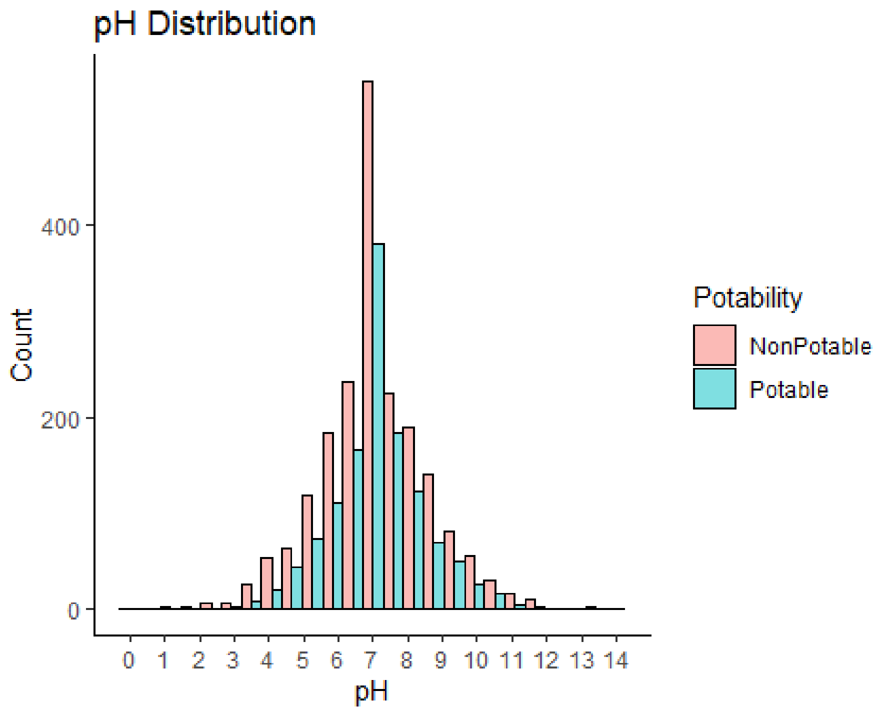
<u>1.ph</u>

<7 is Acidic

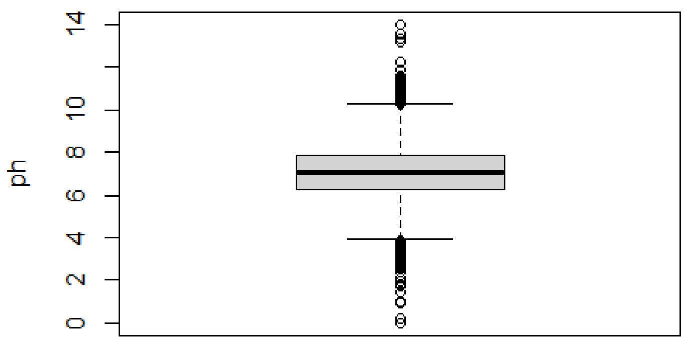
>7 is Basic

The pH of most drinking-water lies within the range 6.5–8.5.

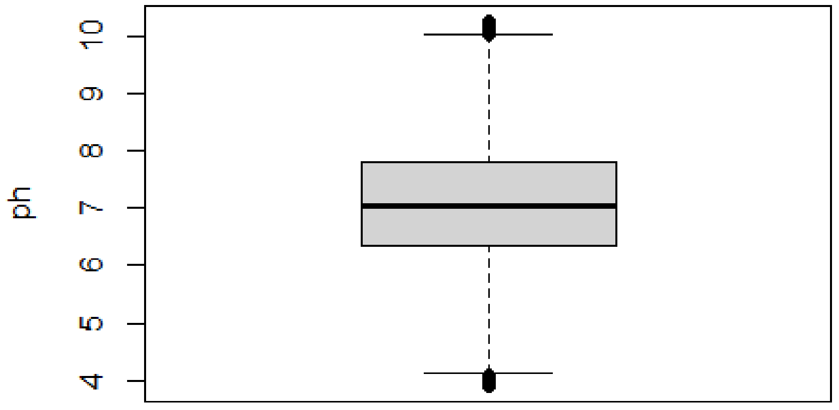
```
ggplot(df_wp,aes(ph,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$ph), max(df_wp$ph), by = 1)
,1)) +
  labs(x = "pH", y = "Count", title = "pH Distribution")
```



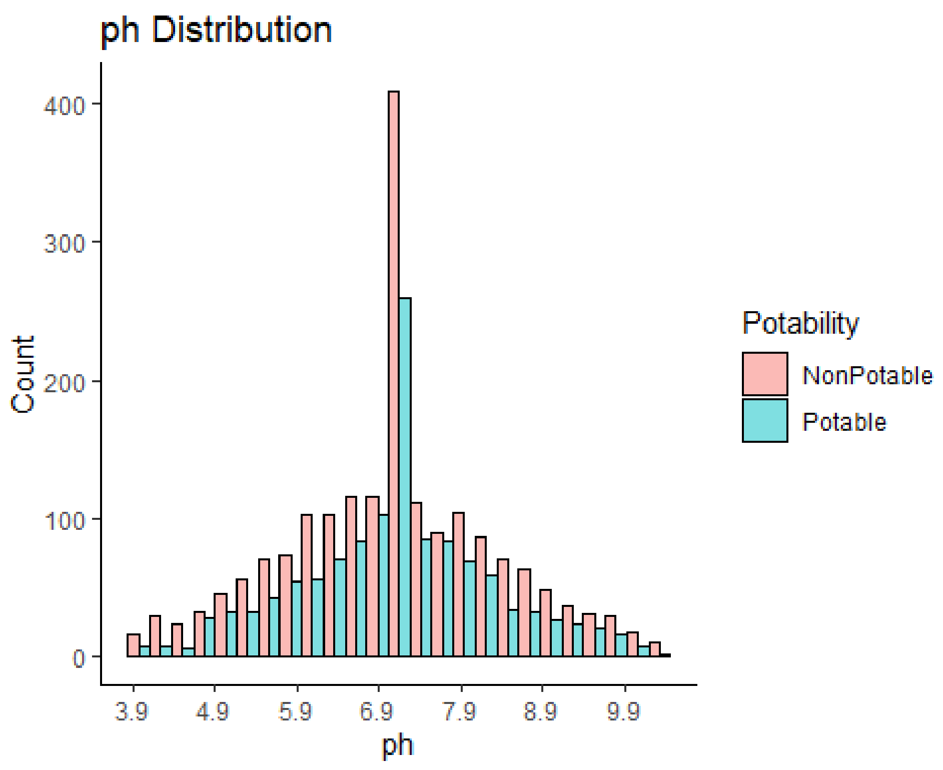
```
boxplot(df_wp$ph,
  ylab = "ph")
```



```
quartiles <- quantile(df_wp$ph, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df_wp$ph)

Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR

df_wp <- subset(df_wp, df_wp$ph > Lower & df_wp$ph < Upper)

dim(df_wp)

## [1] 3134   10

boxplot(df_wp$ph,
  ylab = "ph"
)
```



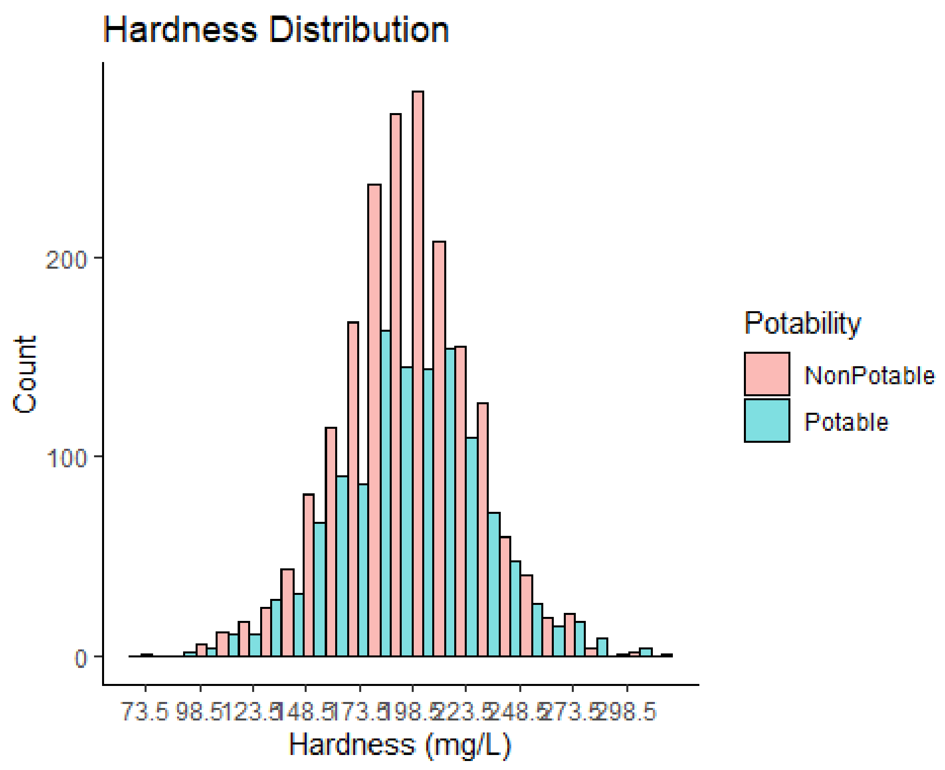
```
ggplot(df_wp,aes(ph,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$ph), max(df_wp$ph), by = 1)
,1)) +
  labs(x = "ph", y = "Count", title = "ph Distribution")
```



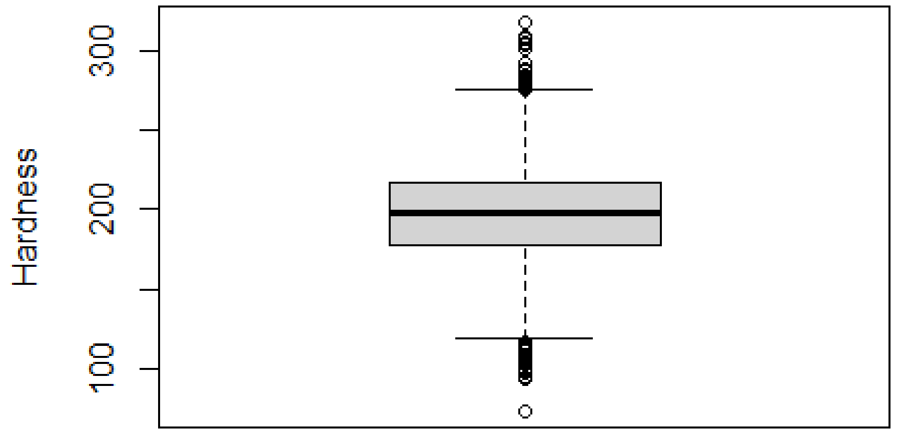
2.Hardness (mg/L)

120 to 170 mg/L considered safe for potable water

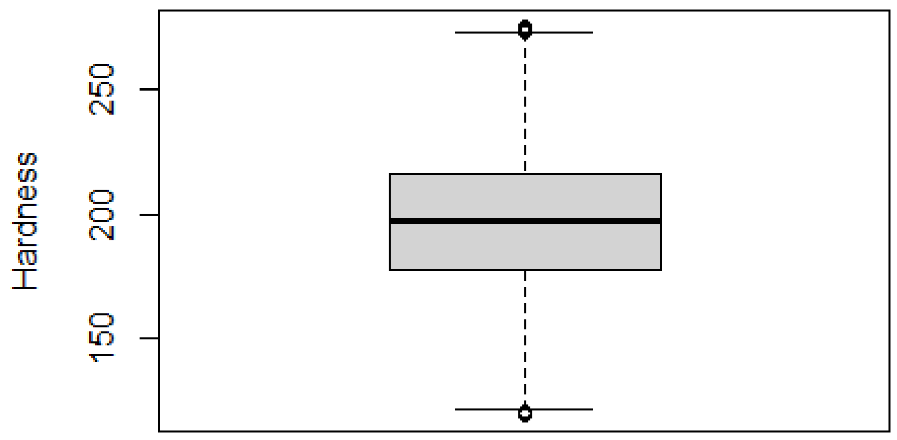
```
ggplot(df_wp,aes(Hardness,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Hardness), max(df_wp$Hardne
ss), by = 25),1)) +
  labs(x = "Hardness (mg/L)", y = "Count", title = "Hardness Distribution")
```



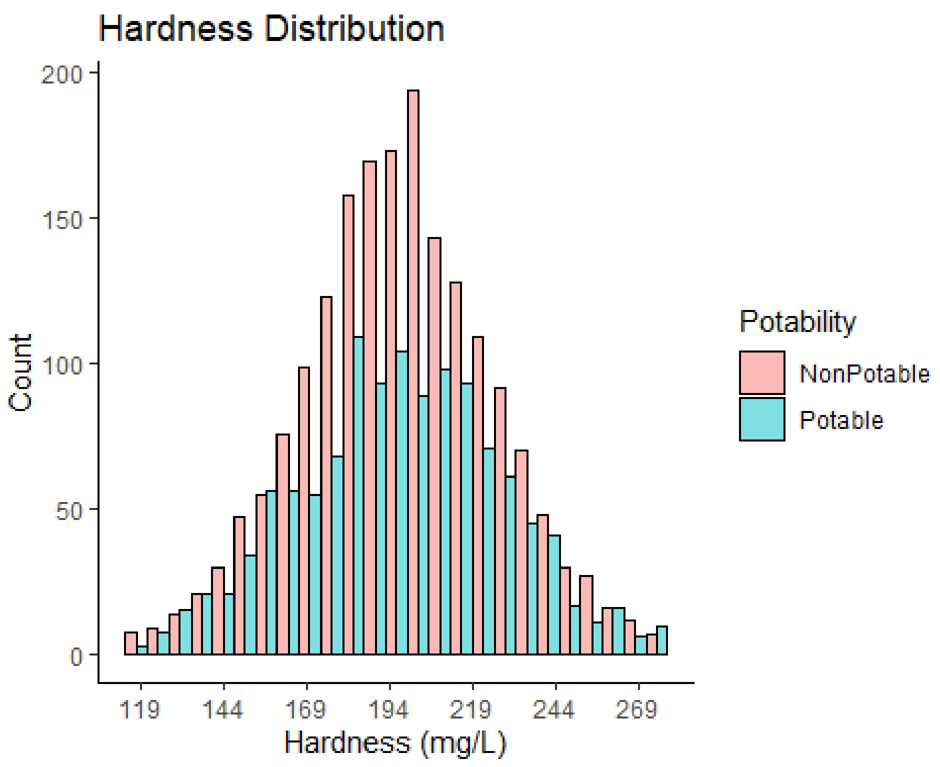
```r
boxplot(df_wp$Hardness,
  ylab = "Hardness"
)
```



```
quartiles <- quantile(df_wp$Hardness, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df_wp$Hardness)

Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR

df_wp <- subset(df_wp, df_wp$Hardness > Lower & df_wp$Hardness < Upper)

dim(df_wp)

## [1] 3060   10

boxplot(df_wp$Hardness,
  ylab = "Hardness"
)
```



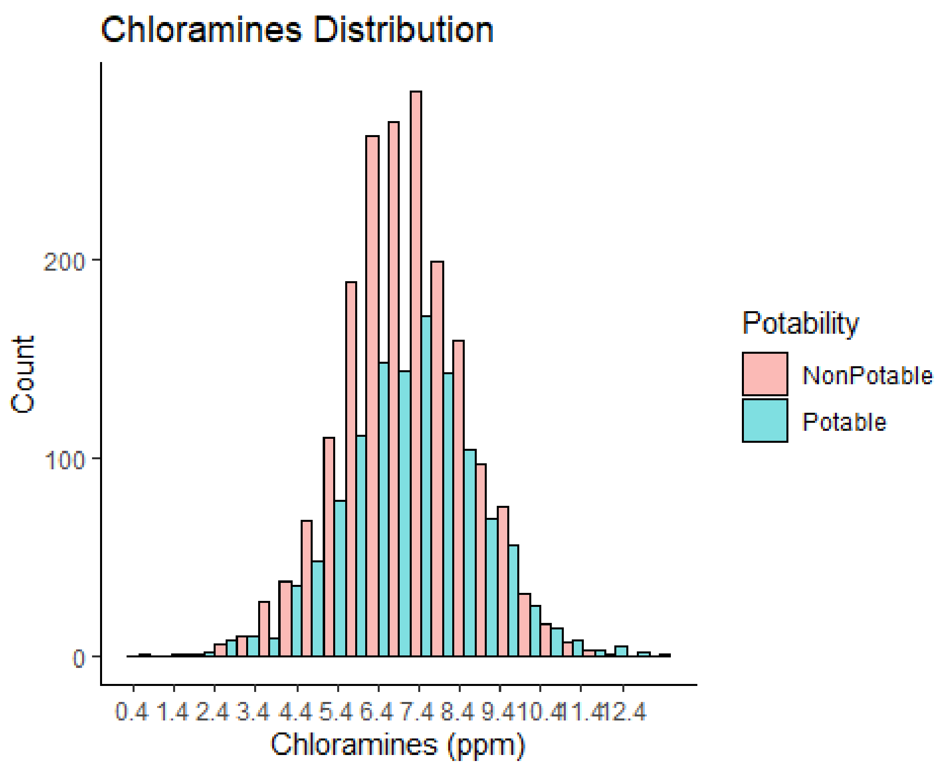
```
ggplot(df_wp,aes(Hardness,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Hardness), max(df_wp$Hardne
ss), by = 25),1)) +
  labs(x = "Hardness (mg/L)", y = "Count", title = "Hardness Distribution")
```



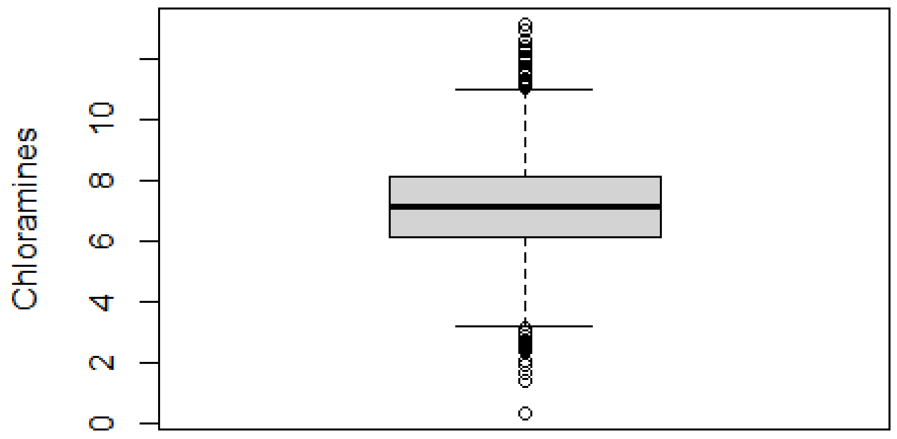
3.Chloramines (ppm)

<4 ppm is considered safe for drinking

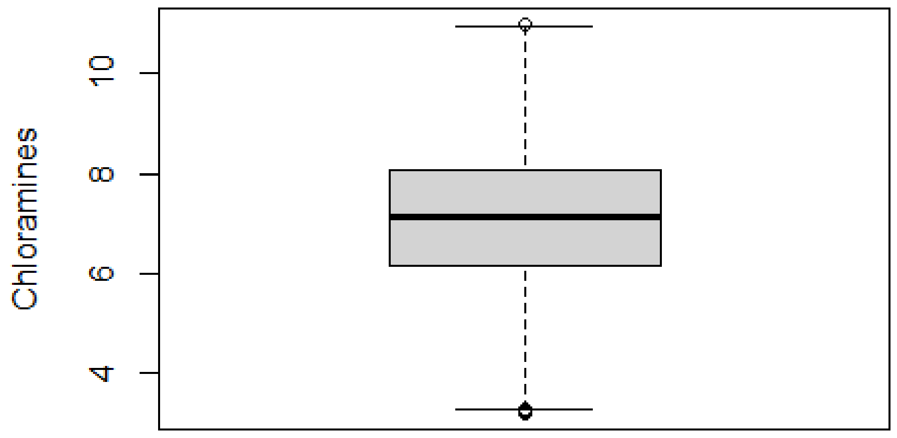
```
ggplot(df_wp,aes(Chloramines,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Chloramines), max(df_wp$Chl
oramines), by = 1),1)) +
  labs(x = "Chloramines (ppm)", y = "Count", title = "Chloramines Distributio
n")
```



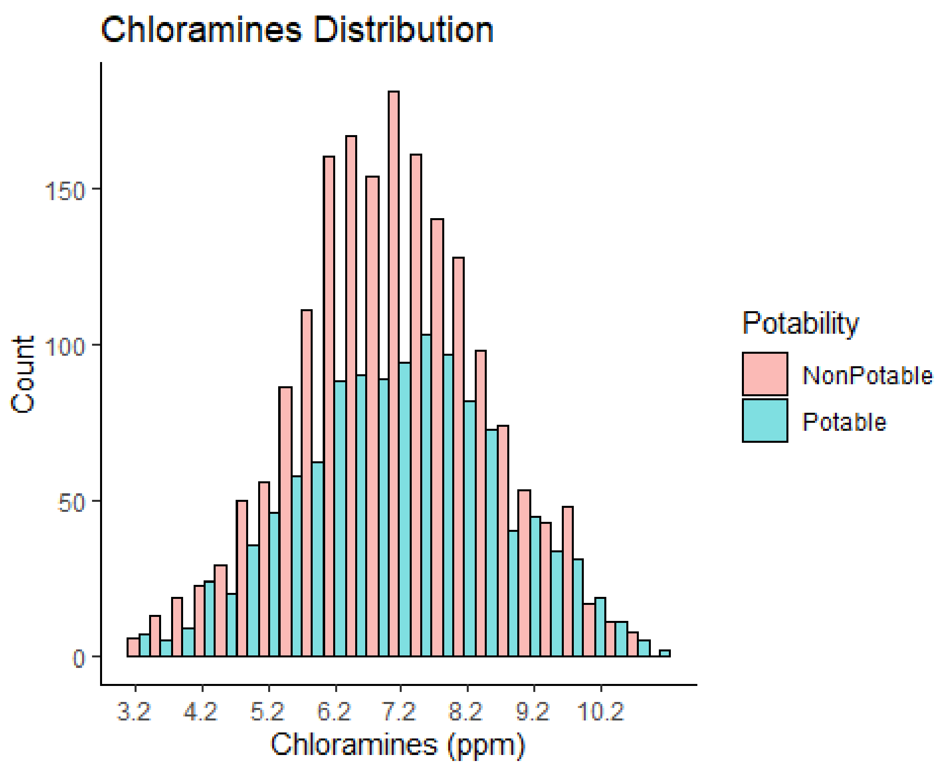
```
boxplot(df_wp$Chloramines,
  ylab = "Chloramines"
)
```



```
quartiles <- quantile(df_wp$Chloramines, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df_wp$Chloramines)

Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR

df_wp <- subset(df_wp, df_wp$Chloramines > Lower & df_wp$Chloramines < Upper)

dim(df_wp)

## [1] 3006   10

boxplot(df_wp$Chloramines,
  ylab = "Chloramines"
)
```



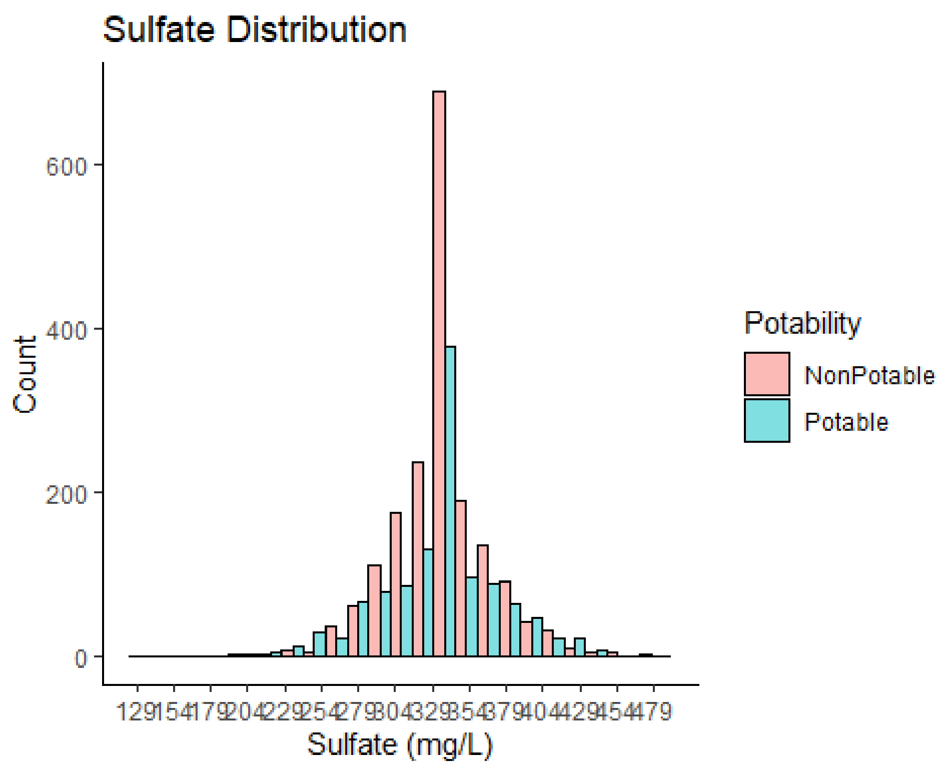
```
ggplot(df_wp,aes(Chloramines,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Chloramines), max(df_wp$Chl
oramines), by = 1),1)) +
  labs(x = "Chloramines (ppm)", y = "Count", title = "Chloramines Distributio
n")
```



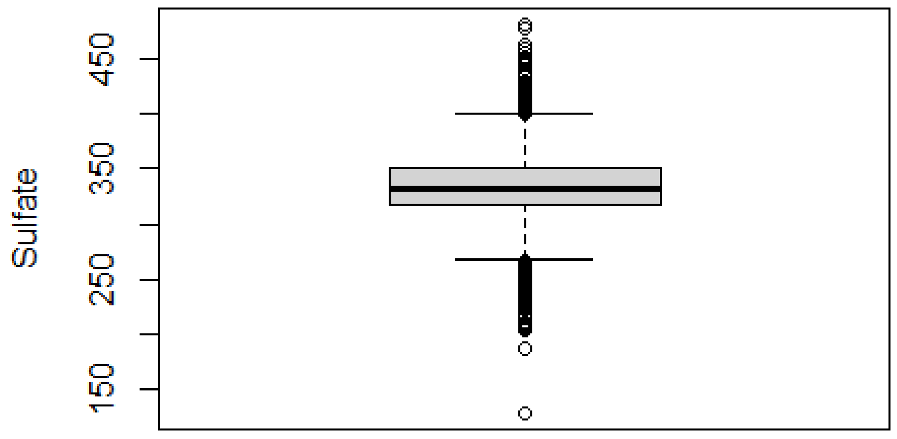
4.Sulfate (mg/L)

<250 mg/L is considered safe for drinking

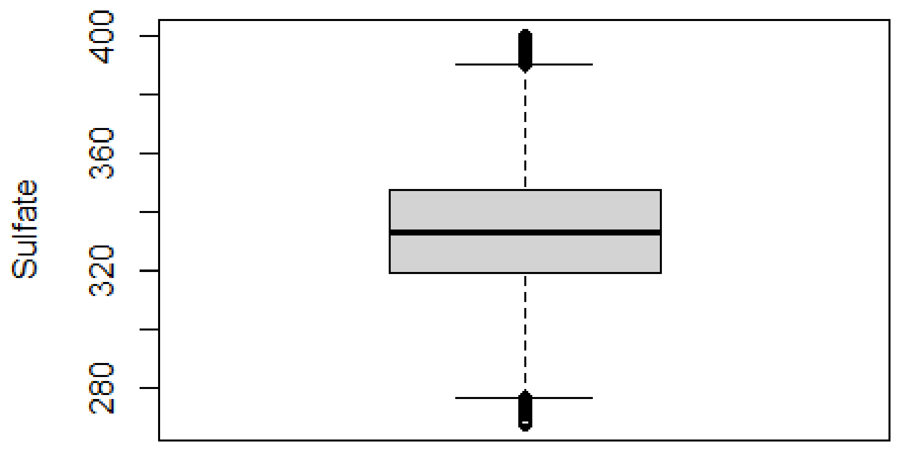
```
ggplot(df_wp,aes(Sulfate,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Sulfate), max(df_wp$Sulfate
), by = 25),1)) +
  labs(x = "Sulfate (mg/L)", y = "Count", title = "Sulfate Distribution")
```



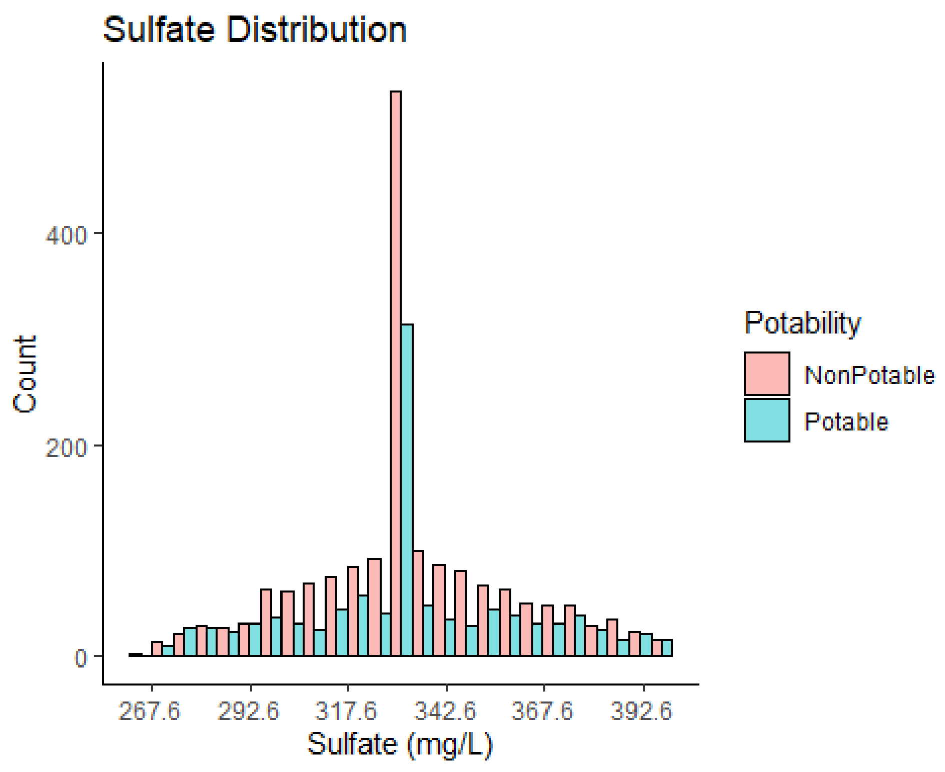
```r
boxplot(df_wp$Sulfate,
  ylab = "Sulfate"
)
```



```
quartiles <- quantile(df_wp$Sulfate, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df_wp$Sulfate)

Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR

df_wp <- subset(df_wp, df_wp$Sulfate > Lower & df_wp$Sulfate < Upper)

dim(df_wp)

## [1] 2773   10

boxplot(df_wp$Sulfate,
  ylab = "Sulfate"
)
```



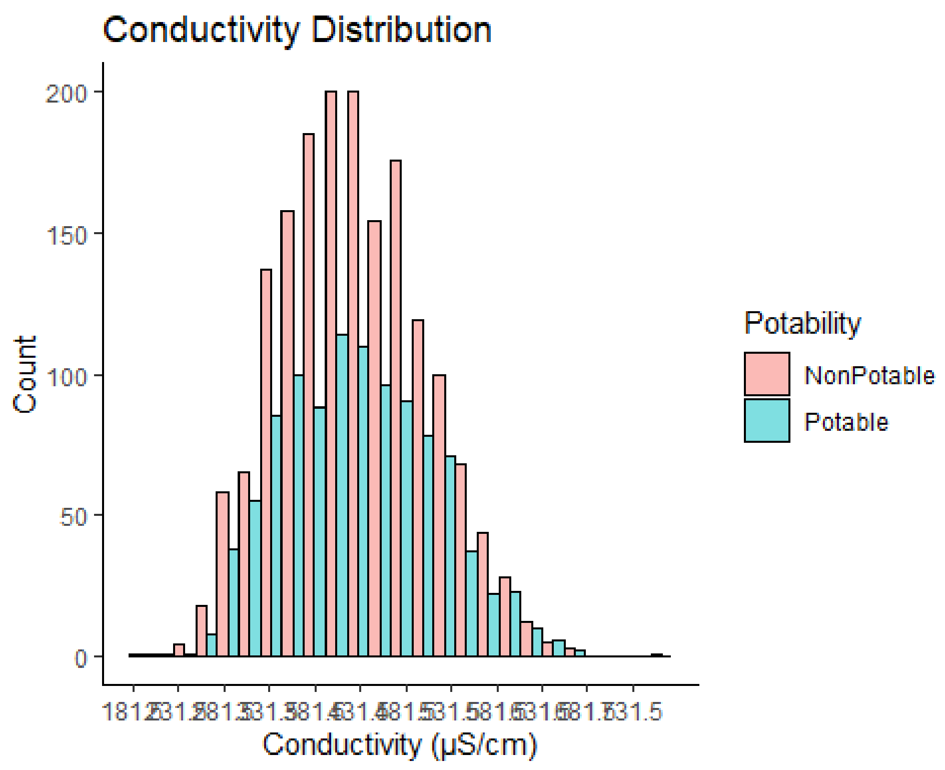
```
ggplot(df_wp,aes(Sulfate,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Sulfate), max(df_wp$Sulfate
), by = 25),1)) +
  labs(x = "Sulfate (mg/L)", y = "Count", title = "Sulfate Distribution")
```



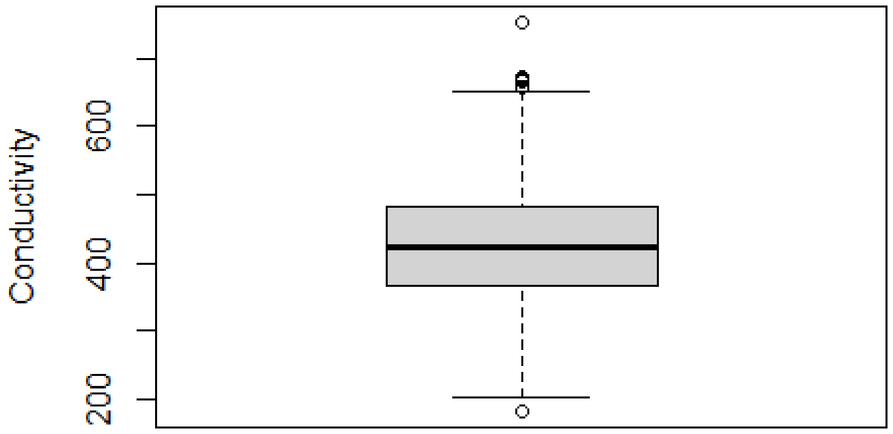
5.Conductivity (µS/cm)

The Conductivity range is safe for both (200-800), Potable and Non-Potable water The limit for drinking water conductivity is 2500 micro-Siemens per centimetre (µS/cm)

```
ggplot(df_wp,aes(Conductivity,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Conductivity), max(df_wp$Co
nductivity), by = 50),1)) +
  labs(x = "Conductivity (µS/cm)", y = "Count", title = "Conductivity Distrib
ution")
```



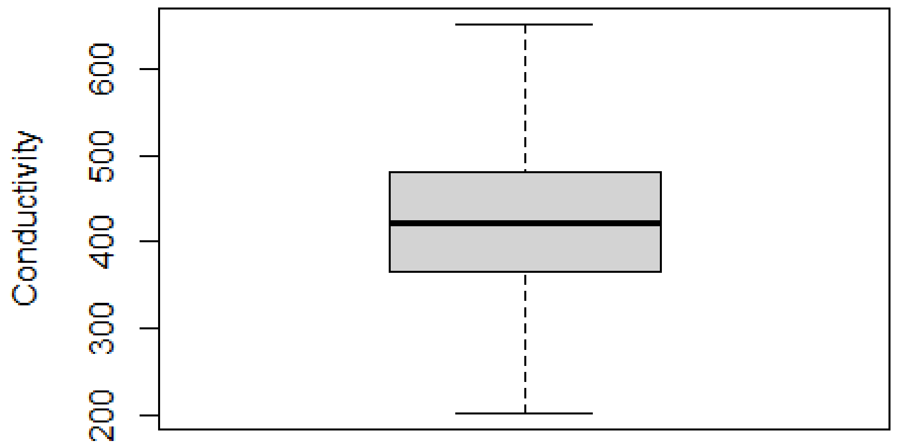
```
boxplot(df_wp$Conductivity,
  ylab = "Conductivity"
)
```



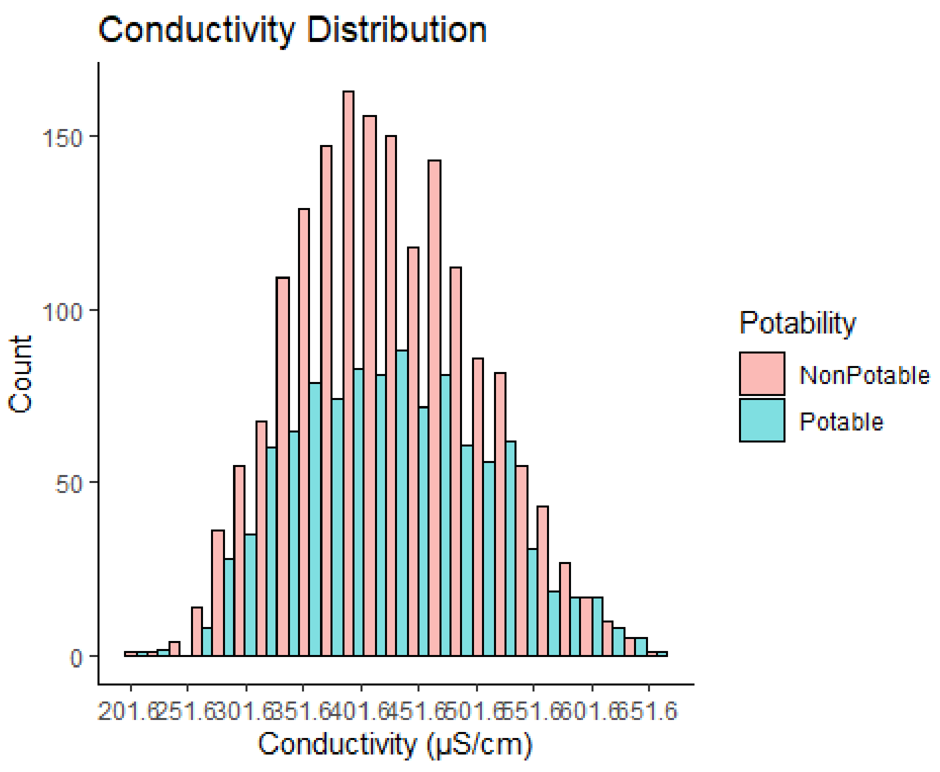
```r
quartiles <- quantile(df_wp$Conductivity, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df_wp$Conductivity)

Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR

df_wp <- subset(df_wp, df_wp$Conductivity > Lower & df_wp$Conductivity < Uppe
r)

dim(df_wp)
```

```
## [1] 2766   10
```

```r
boxplot(df_wp$Conductivity,
  ylab = "Conductivity"
)
```



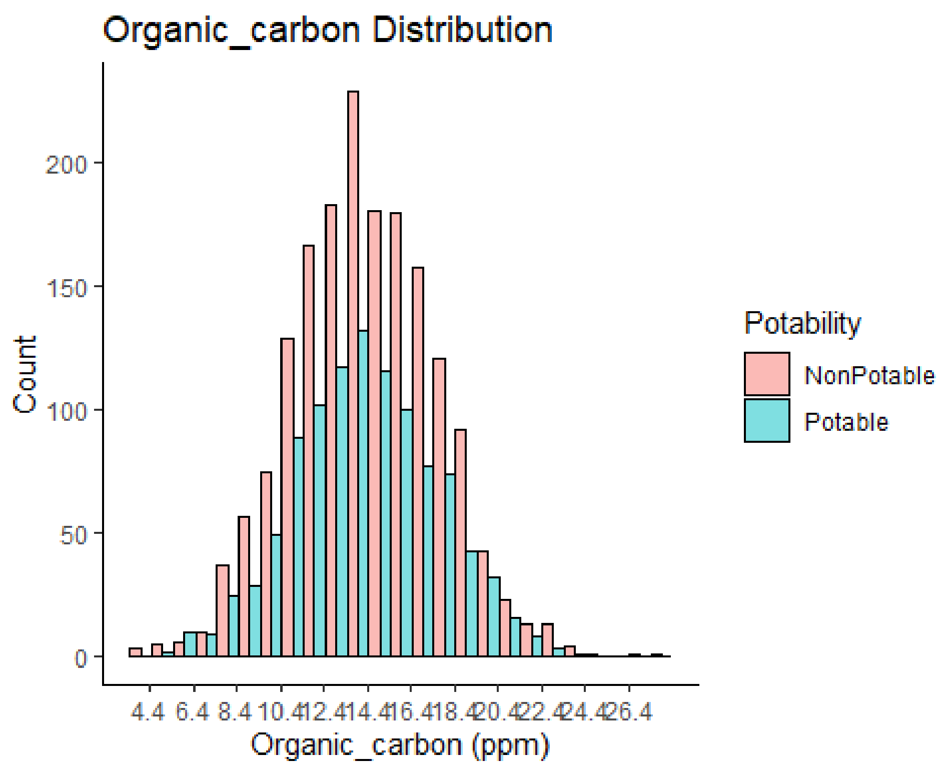
```r
ggplot(df_wp,aes(Conductivity,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Conductivity), max(df_wp$Co
nductivity), by = 50),1)) +
  labs(x = "Conductivity (µS/cm)", y = "Count", title = "Conductivity Distrib
ution")
```



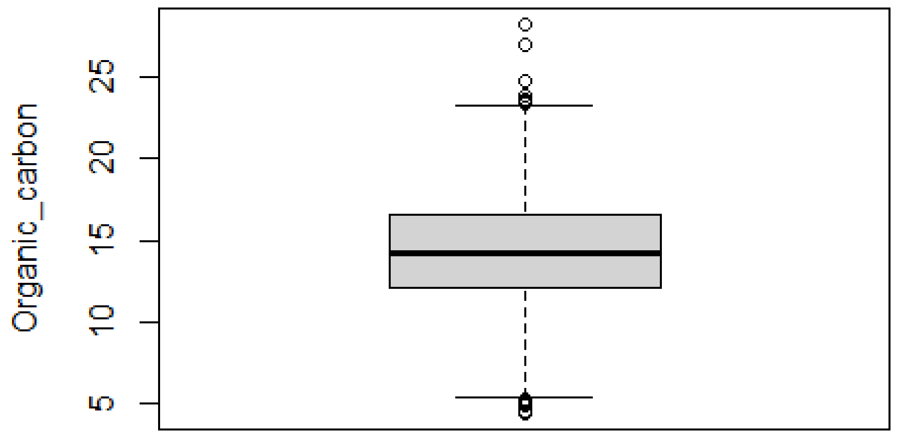
<u>6.Organic_carbon (ppm)</u>

Typical Organic Carbon level is upto 25 ppm

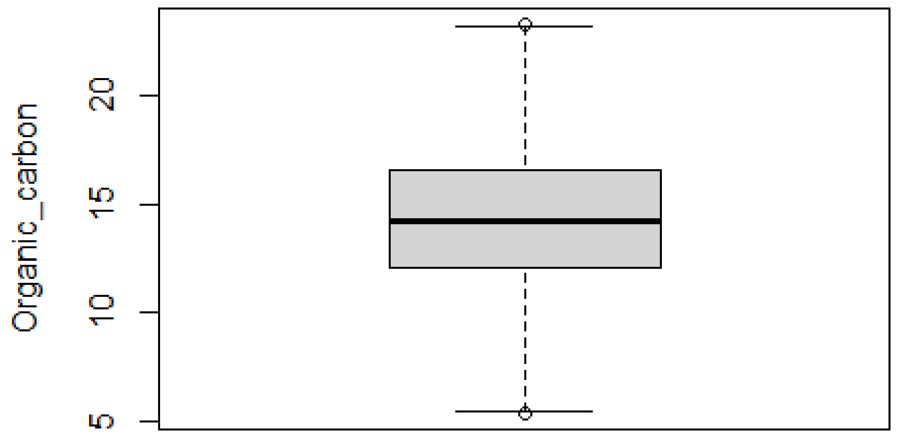
```
ggplot(df_wp,aes(Organic_carbon,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Organic_carbon), max(df_wp$
Organic_carbon), by = 2),1)) +
  labs(x = "Organic_carbon (ppm)", y = "Count", title = "Organic_carbon Distr
ibution")
```



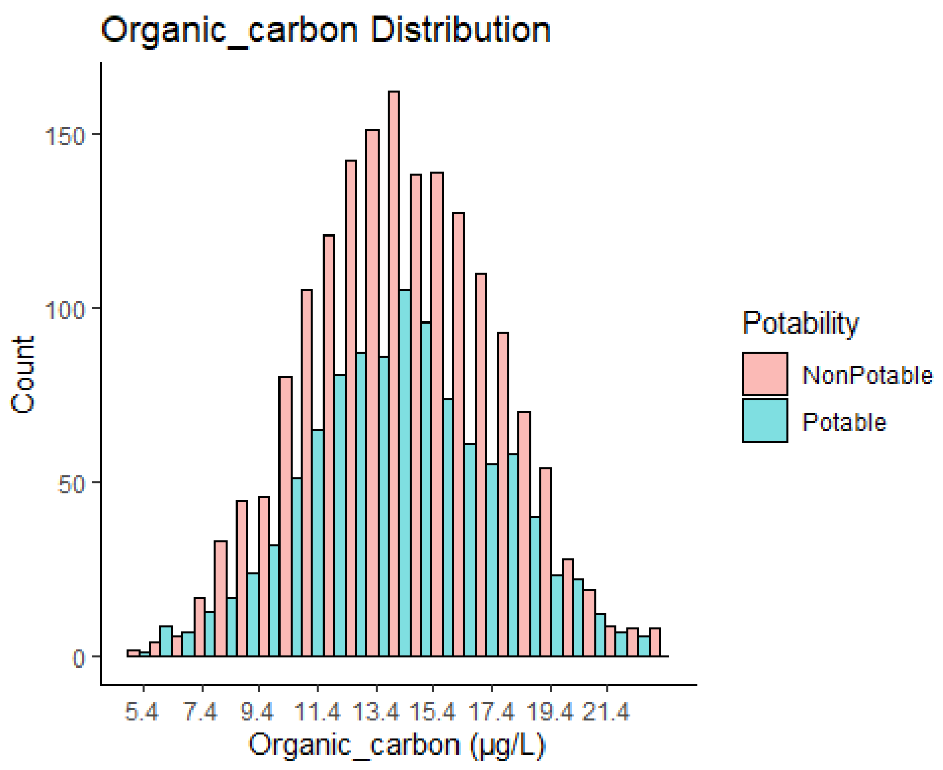
```
boxplot(df_wp$Organic_carbon,
  ylab = "Organic_carbon"
)
```



```
quartiles <- quantile(df_wp$Organic_carbon, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df_wp$Organic_carbon)

Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR

df_wp <- subset(df_wp, df_wp$Organic_carbon > Lower & df_wp$Organic_carbon <
Upper)

dim(df_wp)

## [1] 2749   10

boxplot(df_wp$Organic_carbon,
  ylab = "Organic_carbon"
)
```



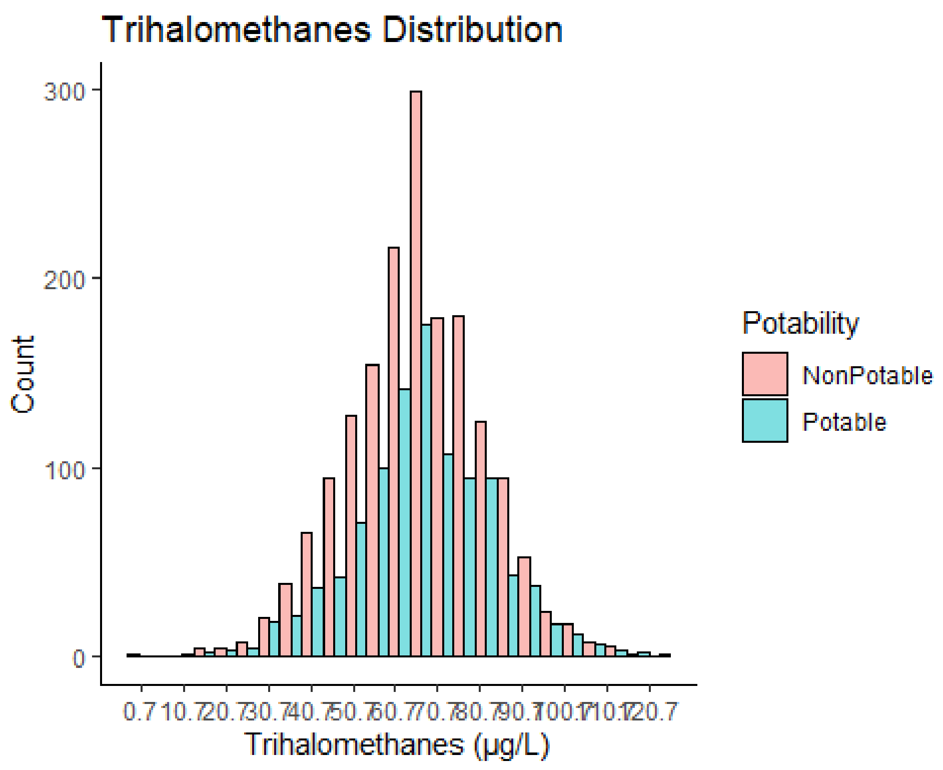
```
ggplot(df_wp,aes(Organic_carbon,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Organic_carbon), max(df_wp$
Organic_carbon), by = 2),1)) +
  labs(x = "Organic_carbon (µg/L)", y = "Count", title = "Organic_carbon Dist
ribution")
```



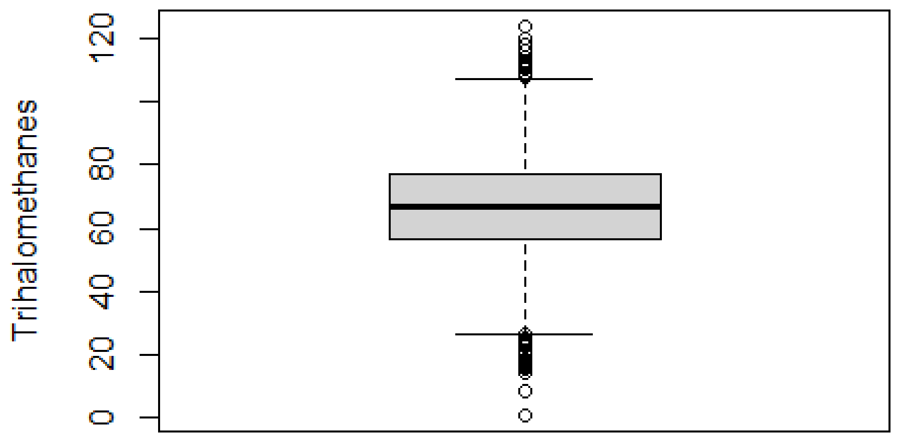
7.Trihalomethanes (µg/L)

Upper limit of Trihalomethanes level is 80 µg/L

```
ggplot(df_wp,aes(Trihalomethanes,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Trihalomethanes), max(df_wp
$Trihalomethanes), by = 10),1)) +
  labs(x = "Trihalomethanes (µg/L)", y = "Count", title = "Trihalomethanes Di
stribution")
```



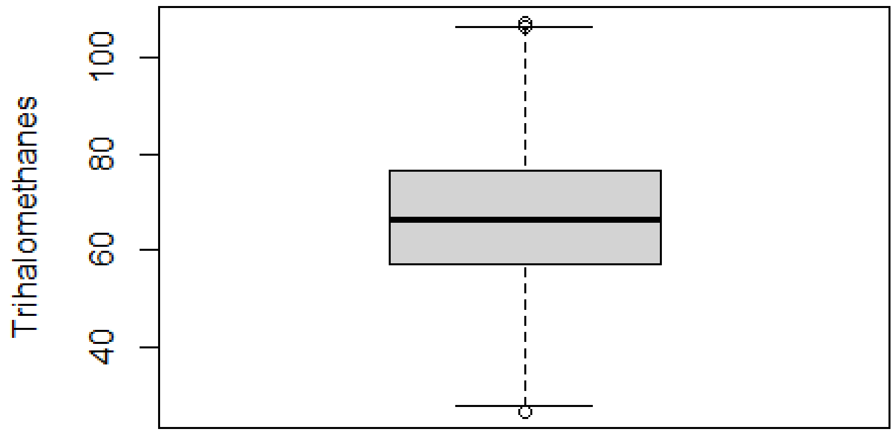
```
boxplot(df_wp$Trihalomethanes,
  ylab = "Trihalomethanes"
)
```



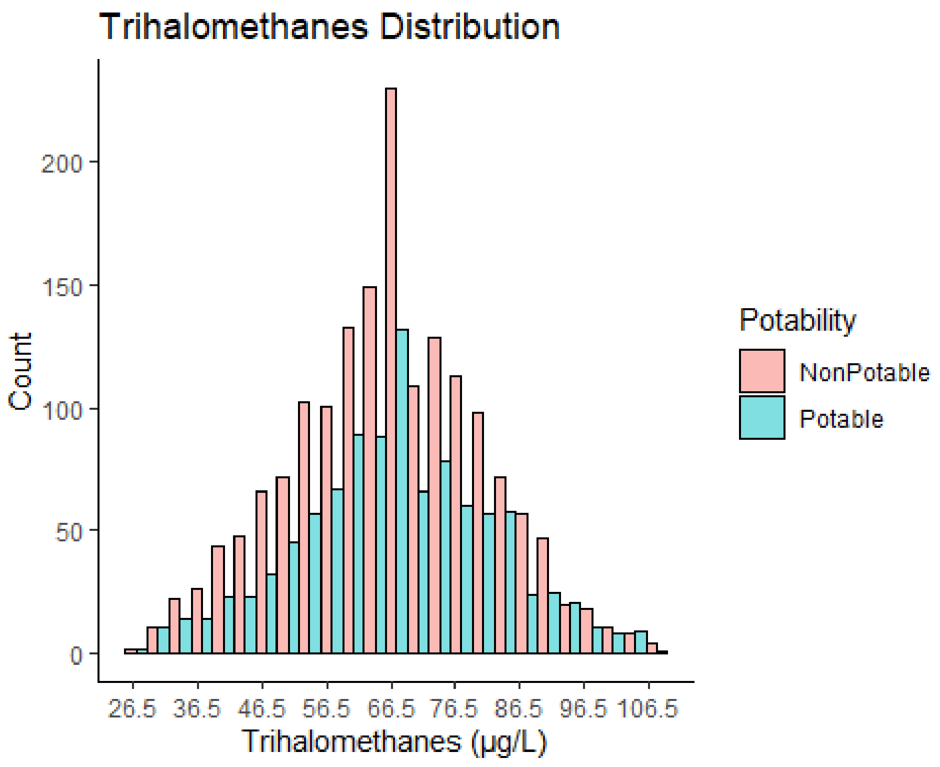
```
quartiles <- quantile(df_wp$Trihalomethanes, probs=c(.25, .75), na.rm = FALSE
)
IQR <- IQR(df_wp$Trihalomethanes)

Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR

df_wp <- subset(df_wp, df_wp$Trihalomethanes > Lower & df_wp$Trihalomethanes
< Upper)

dim(df_wp)
```

```
## [1] 2707   10
```

```
boxplot(df_wp$Trihalomethanes,
  ylab = "Trihalomethanes"
)
```



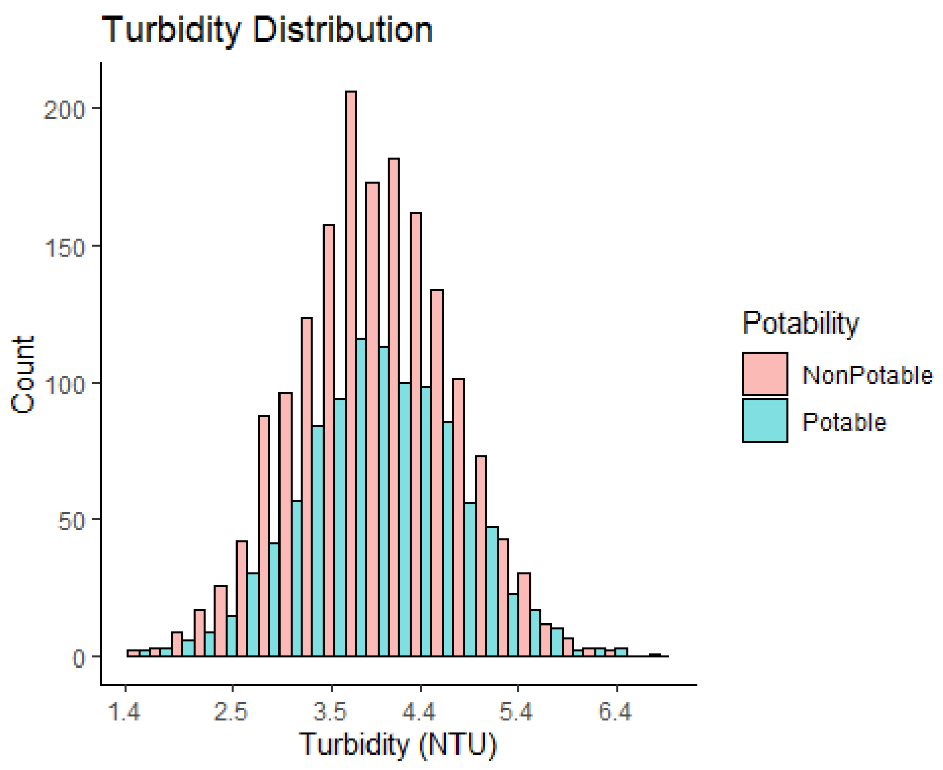
```
ggplot(df_wp,aes(Trihalomethanes,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Trihalomethanes), max(df_wp
$Trihalomethanes), by = 10),1)) +
  labs(x = "Trihalomethanes (µg/L)", y = "Count", title = "Trihalomethanes Di
stribution")
```



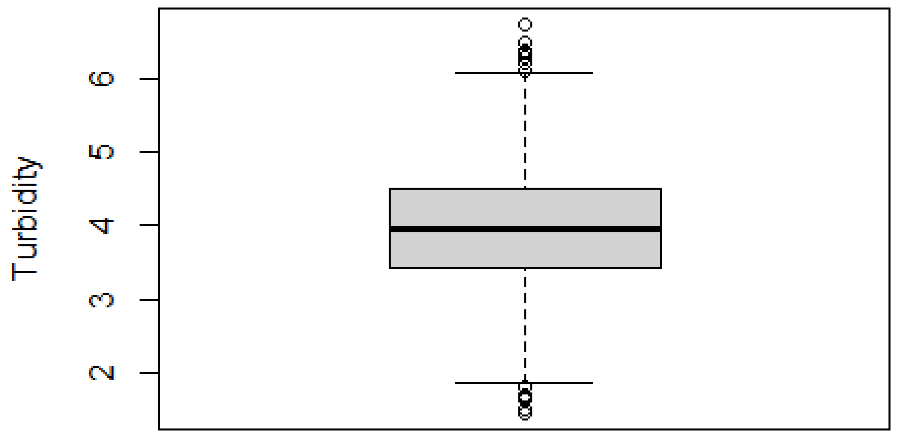
8.Turbidity (NTU)

<5 NTU Turbidity is considered safe

```
ggplot(df_wp,aes(Turbidity,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Turbidity), max(df_wp$Turbi
dity), by = 1),1)) +
  labs(x = "Turbidity (NTU)", y = "Count", title = "Turbidity Distribution")
```



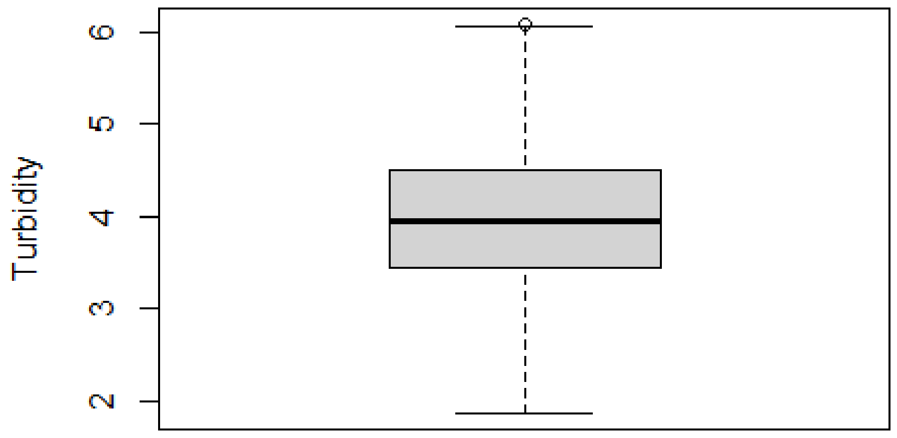
```
boxplot(df_wp$Turbidity,
  ylab = "Turbidity"
)
```



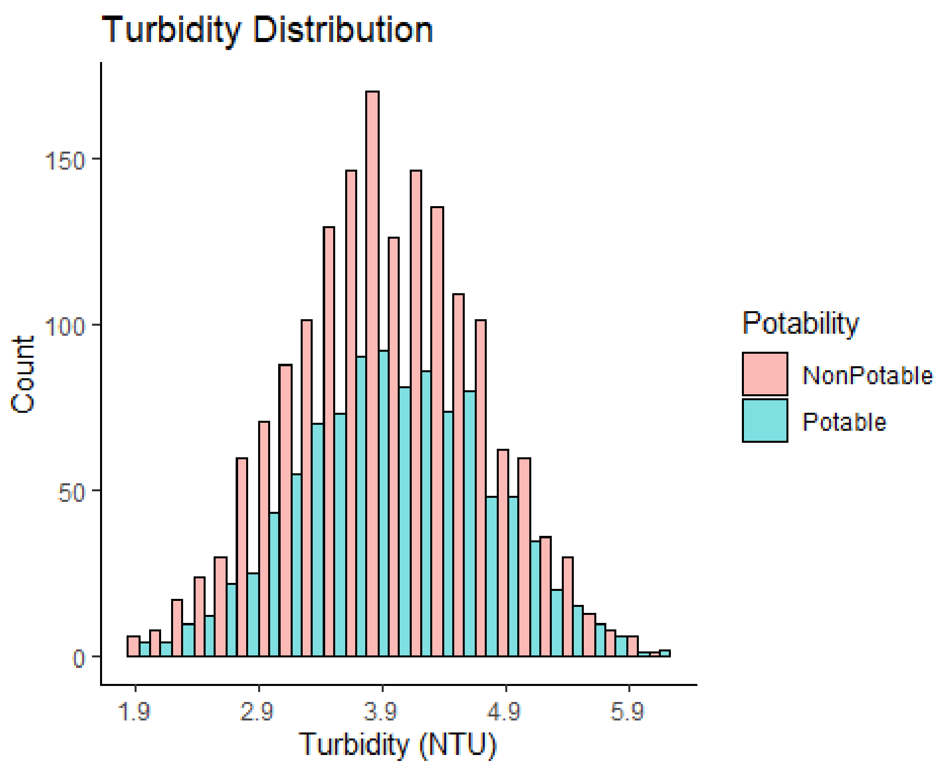
```r
quartiles <- quantile(df_wp$Turbidity, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df_wp$Turbidity)

Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR

df_wp <- subset(df_wp, df_wp$Turbidity > Lower & df_wp$Turbidity < Upper)

dim(df_wp)
```

```
## [1] 2689   10
```

```r
boxplot(df_wp$Turbidity,
  ylab = "Turbidity"
)
```



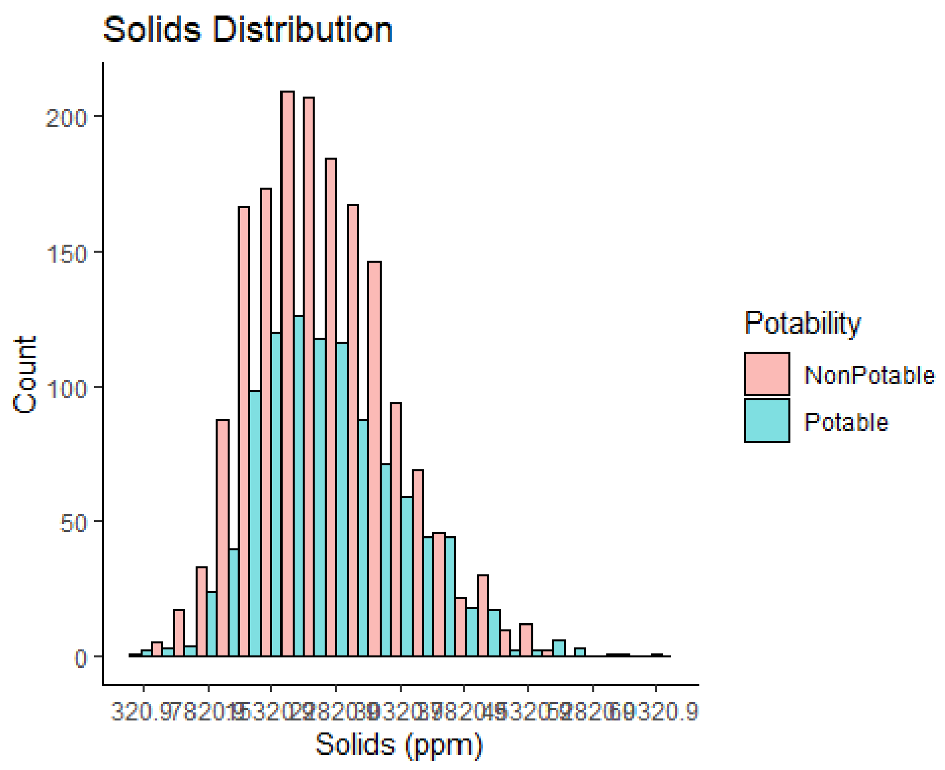
```r
ggplot(df_wp,aes(Turbidity,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Turbidity), max(df_wp$Turbi
dity), by = 1),1)) +
  labs(x = "Turbidity (NTU)", y = "Count", title = "Turbidity Distribution")
```



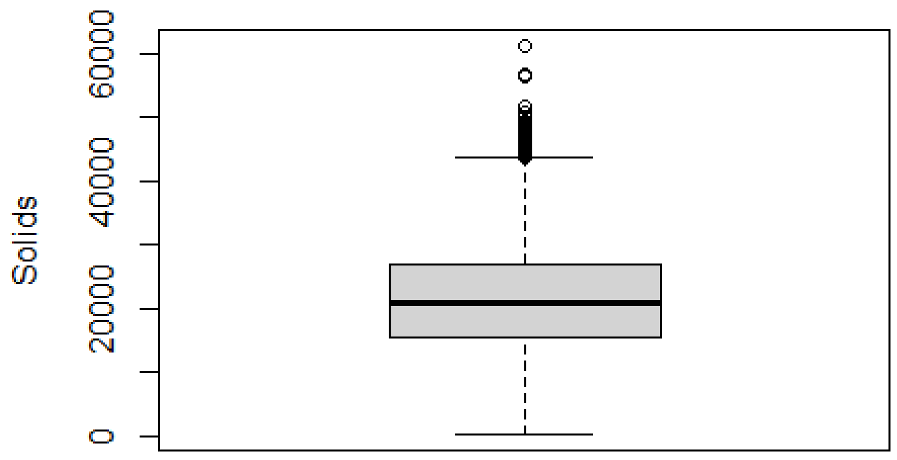
9.Solids (PPM)

<500 ppm is considered safe for drinking

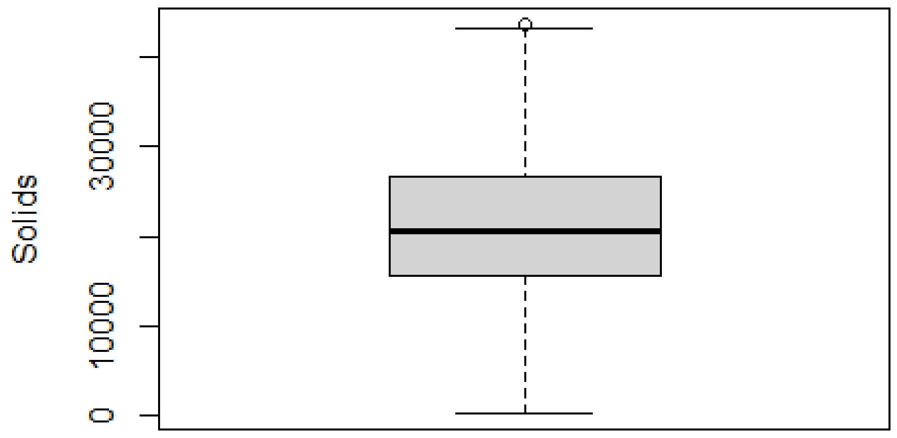
```
ggplot(df_wp,aes(Solids,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Solids), max(df_wp$Solids),
by = 7500),1)) +
  labs(x = "Solids (ppm)", y = "Count", title = "Solids Distribution")
```



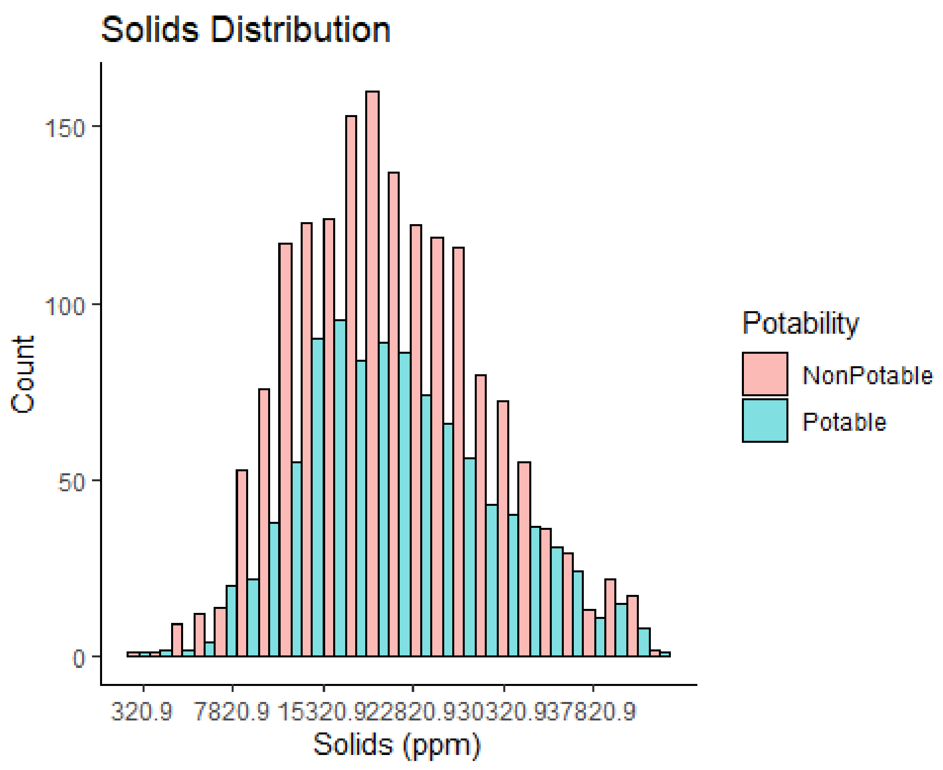
```r
boxplot(df_wp$Solids,
  ylab = "Solids"
)
```



```
quartiles <- quantile(df_wp$Solids, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df_wp$Solids)

Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR

df_wp <- subset(df_wp, df_wp$Solids > Lower & df_wp$Solids < Upper)

dim(df_wp)

## [1] 2657   10

boxplot(df_wp$Solids,
  ylab = "Solids"
)
```



```
ggplot(df_wp,aes(Solids,fill = Potability)) +
  geom_histogram(alpha = 0.5,bins=25,colour="black",position='dodge') +
  theme_classic() +
  scale_x_continuous(breaks = round(seq(min(df_wp$Solids), max(df_wp$Solids),
by = 7500),1)) +
  labs(x = "Solids (ppm)", y = "Count", title = "Solids Distribution")
```



Solids Distribution
Count
150
100
50
0
320.9
7820.9
15320.9
22820.9
30320.9
37820.9
Solids (ppm)
Potability
NonPotable
Potable

AS YOU CAN SEE CLEANING THE DATA HAS PRODUCED A BETTER OUTCOME OF THE DATA, THAT CAN IMPROVE THE MODELS PREDICTION CAPACITY.

```
nrow(df_wp)

## [1] 2657
```

Here we can take a look at the summary after data cleaning has been completed.

```
summary(df_wp)

##        ph            Hardness          Solids         Chloramines
##  Min.   : 3.902   Min.   :119.0   Min.   :  320.9   Min.   : 3.195
##  1st Qu.: 6.353   1st Qu.:178.9   1st Qu.:15548.4   1st Qu.: 6.186
##  Median : 7.037   Median :197.5   Median :20562.5   Median : 7.109
##  Mean   : 7.072   Mean   :196.9   Mean   :21422.8   Mean   : 7.107
##  3rd Qu.: 7.794   3rd Qu.:215.6   3rd Qu.:26687.9   3rd Qu.: 8.051
##  Max.   :10.253   Max.   :275.7   Max.   :43680.2   Max.   :11.000
##     Sulfate       Conductivity   Organic_carbon   Trihalomethanes
##  Min.   :267.6   Min.   :201.6   Min.   : 5.362   Min.   : 26.51
##  1st Qu.:319.5   1st Qu.:365.6   1st Qu.:12.087   1st Qu.: 56.96
##  Median :333.1   Median :421.1   Median :14.236   Median : 66.62
##  Mean   :333.6   Mean   :425.8   Mean   :14.314   Mean   : 66.50
##  3rd Qu.:348.0   3rd Qu.:481.3   3rd Qu.:16.576   3rd Qu.: 76.69
##  Max.   :400.0   Max.   :652.5   Max.   :23.318   Max.   :107.31
##    Turbidity      Potability
##  Min.   :1.873   Length:2657
##  1st Qu.:3.439   Class :character
##  Median :3.947   Mode  :character
##  Mean   :3.962
##  3rd Qu.:4.497
##  Max.   :6.084
```

D.PREPROCESSING

Preprocessing is necessary to standardize the data.

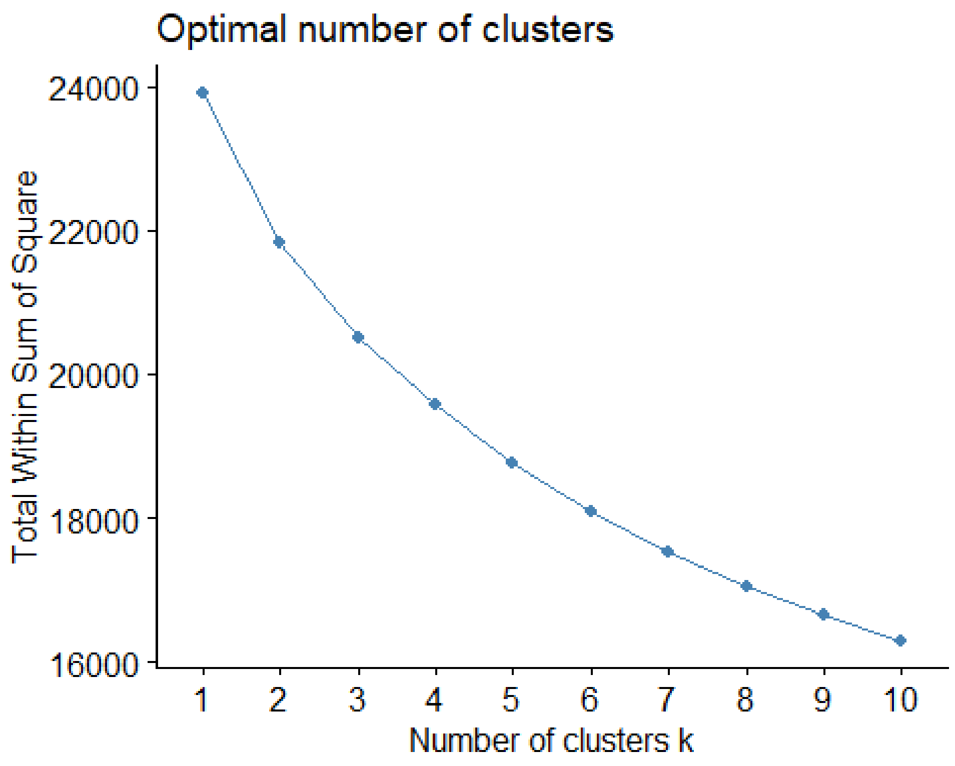
```
# Remove class labels
predictors <- df_wp %>% select(-c(Potability))
head(predictors)

##          ph Hardness   Solids Chloramines  Sulfate Conductivity Organic_ca
rbon
## 1  7.036752 204.8905 20791.32    7.300212 368.5164     564.3087       10.37
9783
## 3  8.099124 224.2363 19909.54    9.275884 333.0735     418.6062       16.86
8637
## 4  8.316766 214.3734 22018.42    8.059332 356.8861     363.2665       18.43
6524
## 5  9.092223 181.1015 17978.99    6.546600 310.1357     398.4108       11.55
8279
## 6  5.584087 188.3133 28748.69    7.544869 326.6784     280.4679        8.39
9735
## 7 10.223862 248.0717 28749.72    7.513408 393.6634     283.6516       13.78
9695
##   Trihalomethanes Turbidity
## 1        86.99097  2.963135
## 3        66.42009  3.055934
## 4       100.34167  4.628771
## 5        31.99799  4.075075
## 6        54.91786  2.559708
## 7        84.60356  2.672989

# Center scale allows us to standardize the data
preproc <- preProcess(predictors, method=c("center", "scale"))
# We have to call predict to fit our data based on preprocessing
predictors <- predict(preproc, predictors)
#data has been standardized
summary(predictors)
```

```
##        ph               Hardness            Solids          Chloramines
##  Min.   :-2.60905   Min.   :-2.75441   Min.   :-2.6947   Min.   :-2.771000
##  1st Qu.:-0.59150   1st Qu.:-0.63626   1st Qu.:-0.7502   1st Qu.:-0.652531
##  Median :-0.02901   Median : 0.02035   Median :-0.1099   Median : 0.001141
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.000000
##  3rd Qu.: 0.59426   3rd Qu.: 0.66128   3rd Qu.: 0.6724   3rd Qu.: 0.668609
##  Max.   : 2.61836   Max.   : 2.78326   Max.   : 2.8423   Max.   : 2.757235
##     Sulfate         Conductivity       Organic_carbon     Trihalomethanes
##  Min.   :-2.4982   Min.   :-2.80458   Min.   :-2.78002   Min.   :-2.671804
##  1st Qu.:-0.5344   1st Qu.:-0.75234   1st Qu.:-0.69165   1st Qu.:-0.637611
##  Median :-0.0210   Median :-0.05769   Median :-0.02413   Median : 0.008059
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.000000
##  3rd Qu.: 0.5425   3rd Qu.: 0.69534   3rd Qu.: 0.70266   3rd Qu.: 0.680887
##  Max.   : 2.5090   Max.   : 2.83783   Max.   : 2.79637   Max.   : 2.725789
##    Turbidity
##  Min.   :-2.74442
##  1st Qu.:-0.68720
##  Median :-0.01989
##  Mean   : 0.00000
##  3rd Qu.: 0.70175
##  Max.   : 2.78614
```

E. Clustering

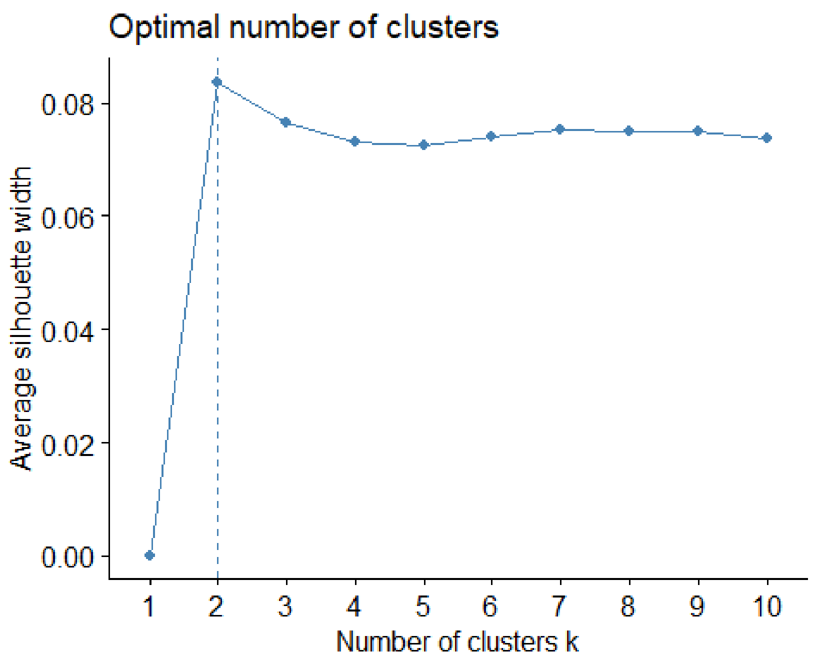
The goal is to group the objects in a set so that they are more similar to one another than to the objects in other groups.

```
set.seed(123)

# Find the knee
fviz_nbclust(predictors, kmeans, method = "wss")
```



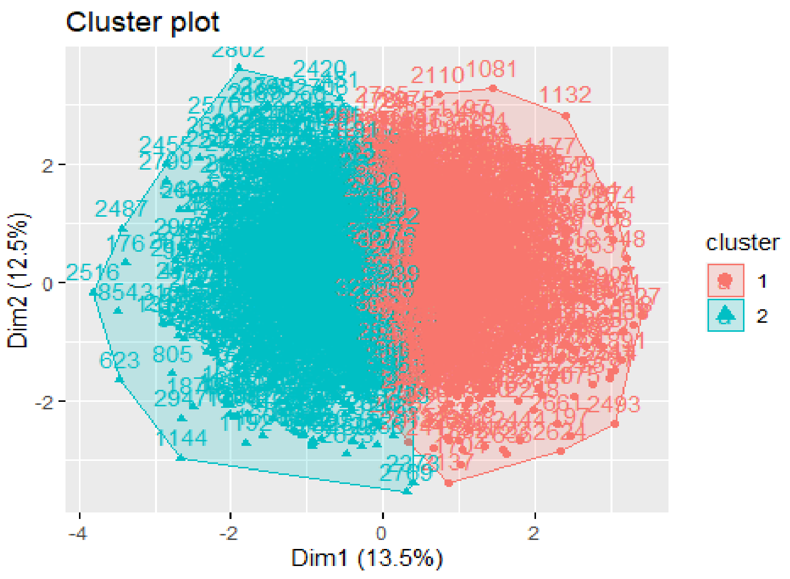
The plot depicts flattening from cluster 3, therefore K=2 is ideal.

```
fviz_nbclust(predictors, kmeans, method = "silhouette")
```



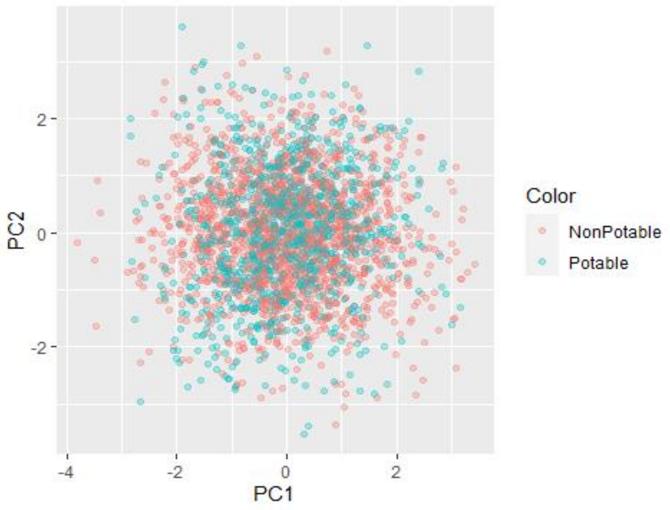
The Silhouette plot also suggest K=2.

```
# Fit the data
fit <- kmeans(predictors, centers = 2, nstart = 25)


# Display the cluster plot
fviz_cluster(fit, data = predictors)
```



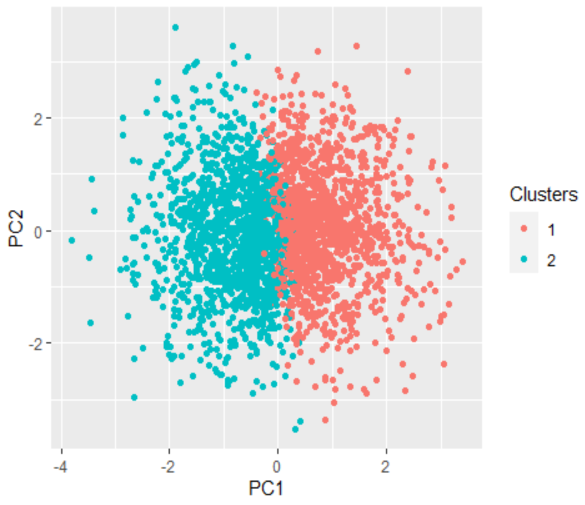
The cluster plot reveals two distinct groupings with slight convergence in the middle.

```
# Calculate PCA
pca = prcomp(predictors)
# Save as dataframe
rotated_data = as.data.frame(pca$x)
# Add original labels as a reference
rotated_data$Color <- df_wp$Potability
# Plot and color by labels
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Color)) + geom_point(
alpha = 0.3)
```



Using PCA we can now check the clustering with the target variable, showing as that there is a discrepancy in grouping revealing that the features are similar in most cases of potable and non-potable.

```
# Assign clusters as a new column
rotated_data$Clusters = as.factor(fit$cluster)
# Plot and color by labels
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Clusters)) + geom_poi
nt()
```

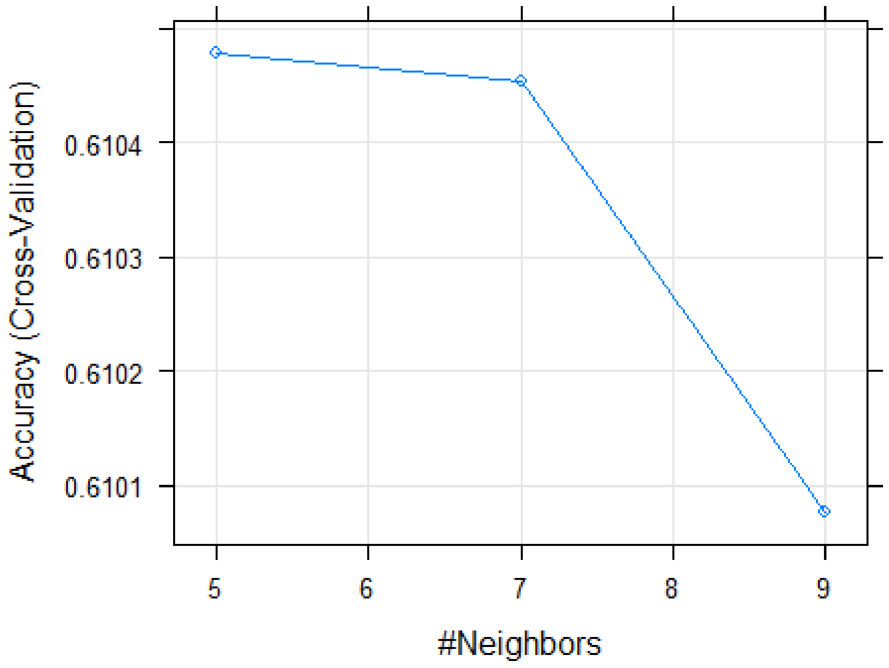


F.Classification

1.knn: The k-nearest neighbors algorithm, sometimes referred to as KNN or k-NN, is a non-parametric, supervised learning classifier that relies on closeness to produce classifications or predictions about the grouping of a single data point.

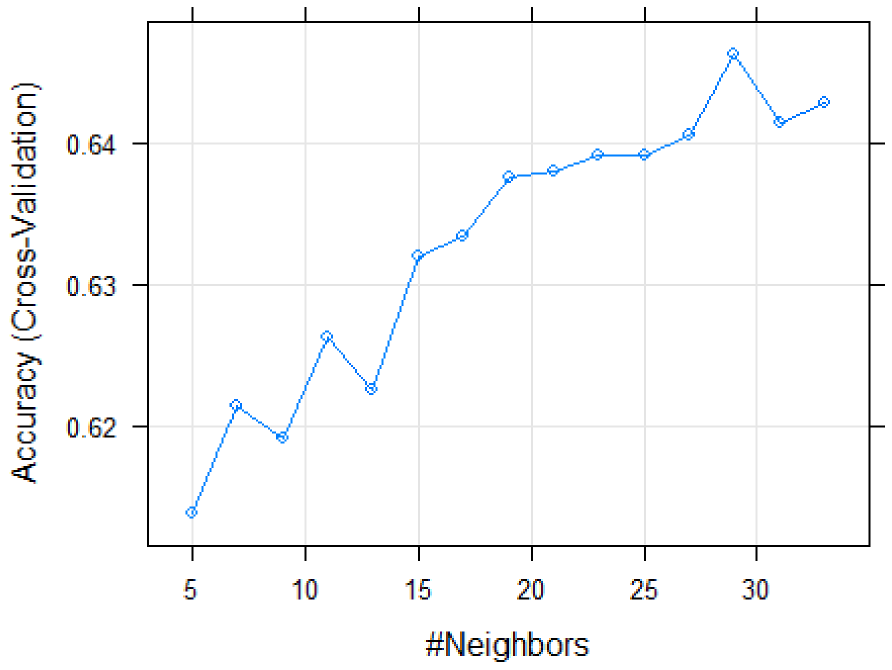
Here we perform knn by applying Tunelength and Tunegrid, to see how the model performs, and which is better.

```
set.seed(123)
#knn

# Remember scaling is crucial for KNN
ctrl <- trainControl(method="cv", number = 10)
knnFit <- train(Potability~ ., data = df_wp,
 method = "knn",
 trControl = ctrl,
 preProcess = c("center","scale"))
knnFit

## k-Nearest Neighbors
##
## 2657 samples
##    9 predictor
##    2 classes: 'NonPotable', 'Potable'
##
## Pre-processing: centered (9), scaled (9)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2391, 2392, 2392, 2392, 2392, 2391, ...
## Resampling results across tuning parameters:
##
##    k  Accuracy   Kappa
##    5  0.6104784  0.11776004
##    7  0.6104529  0.10559851
##    9  0.6100770  0.08904955
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.

plot(knnFit)
```



```
df_wp$Potability <- as.factor(df_wp$Potability)

#APPLYING THE MODEL
# Predict
pred_knn <- predict(knnFit, df_wp)
# Generate confusion matrix
confusionMatrix(df_wp$Potability, pred_knn)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction    NonPotable Potable
##   NonPotable        1457     206
##   Potable            460     534
##
##                Accuracy : 0.7493
##                  95% CI : (0.7324, 0.7657)
##     No Information Rate : 0.7215
##     P-Value [Acc > NIR] : 0.0006594
##
##                   Kappa : 0.4357
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7600
##             Specificity : 0.7216
##          Pos Pred Value : 0.8761
##          Neg Pred Value : 0.5372
##              Prevalence : 0.7215
##          Detection Rate : 0.5484
##    Detection Prevalence : 0.6259
##       Balanced Accuracy : 0.7408
##
##        'Positive' Class : NonPotable
##

#knn tuning performed

# Remember scaling is crucial for KNN
ctrl <- trainControl(method="cv", number = 10)
knnFit2 <- train(Potability~ ., data = df_wp,
 method = "knn",
 trControl = ctrl,
 preProcess = c("center","scale"),tuneLength = 15)
knnFit2

## k-Nearest Neighbors
##
## 2657 samples
##    9 predictor
##    2 classes: 'NonPotable', 'Potable'
##
## Pre-processing: centered (9), scaled (9)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2391, 2391, 2392, 2391, 2391, 2391, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.6138488  0.12564248
##    7  0.6213662  0.12819873
##    9  0.6191247  0.11198142
##   11  0.6262761  0.12063475
##   13  0.6225110  0.09979408
##   15  0.6319152  0.11216445
##   17  0.6334260  0.10724854
##   19  0.6375670  0.11340724
##   21  0.6379387  0.10785176
##   23  0.6390736  0.10358682
##   25  0.6390708  0.09665981
##   27  0.6405689  0.09881692
##   29  0.6462222  0.10941647
##   31  0.6413278  0.09340735
##   33  0.6428302  0.09372169
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 29.
```

```
plot(knnFit2)
```

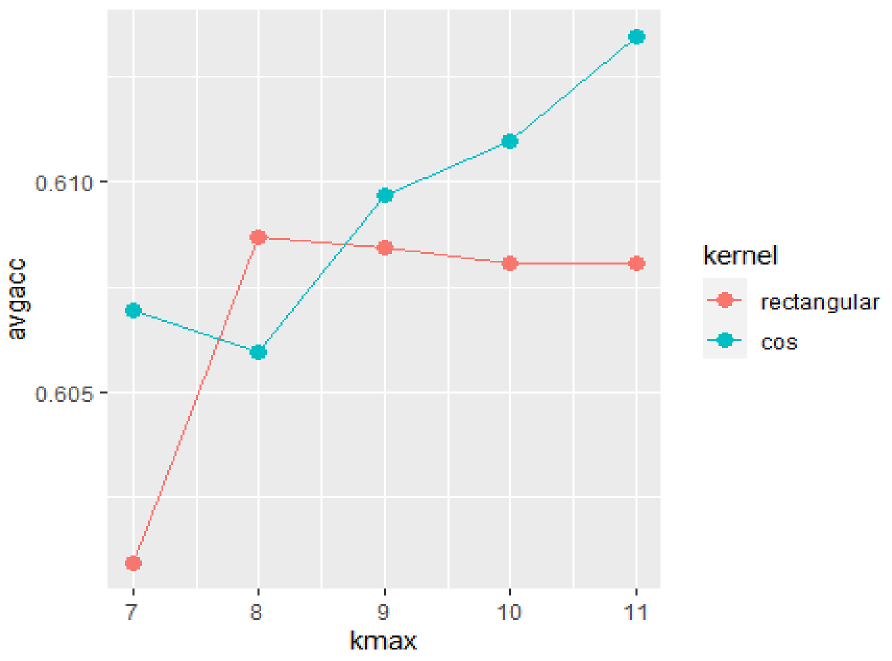


```
#APPLYING THE MODEL
# Predict
pred_knn <- predict(knnFit2, df_wp)
# Generate confusion matrix
confusionMatrix(df_wp$Potability, pred_knn)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   NonPotable Potable
##   NonPotable       1584      79
##   Potable           803     191
##
##                Accuracy : 0.668
##                  95% CI : (0.6498, 0.6859)
##     No Information Rate : 0.8984
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1695
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6636
##             Specificity : 0.7074
##          Pos Pred Value : 0.9525
##          Neg Pred Value : 0.1922
##              Prevalence : 0.8984
##          Detection Rate : 0.5962
##    Detection Prevalence : 0.6259
##       Balanced Accuracy : 0.6855
##
##        'Positive' Class : NonPotable
##
```

The final value used for the model was k = 9. Lets try setting a grid, draw comparison of accuracy.

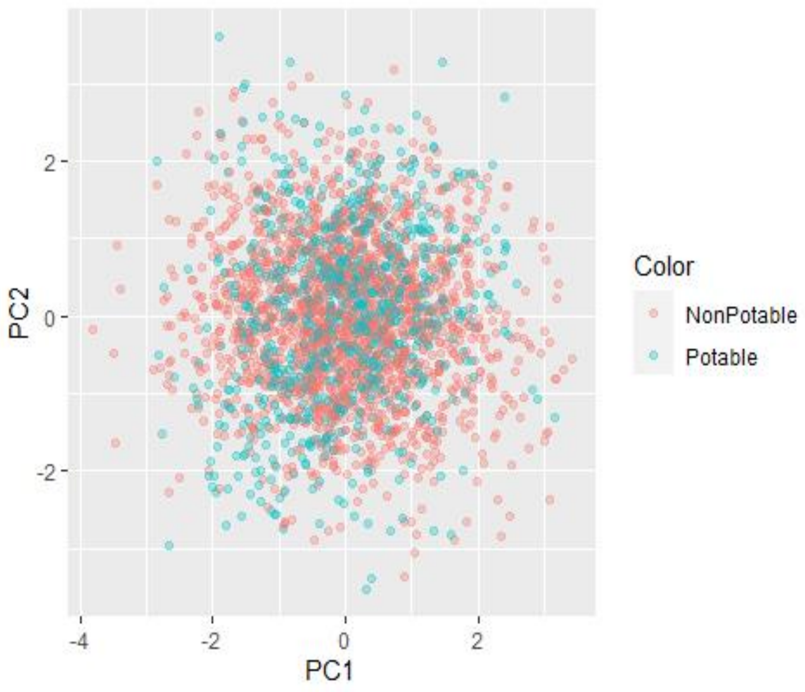
```
set.seed(123)

# setup a tuneGrid with the tuning parameters
tuneGrid <- expand.grid(kmax = 7:11,      # test a range of k values 8 to 10
 kernel = c("rectangular", "cos"),      # regular and cosine-based distance f
unctions
 distance = 1:3)                        # powers of Minkowski 1 to 3

# tune and fit the model with 10-fold cross validation,

# standardization, and our specialized tune grid

kknn_fit <- train(Potability ~ .,
 data = df_wp,
 method = 'kknn',
 trControl = ctrl,
 preProcess = c('center', 'scale'),
 tuneGrid = tuneGrid)
# Printing trained model provides report
kknn_fit

## k-Nearest Neighbors
##
## 2657 samples
##    9 predictor
##    2 classes: 'NonPotable', 'Potable'
##
## Pre-processing: centered (9), scaled (9)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2391, 2392, 2392, 2392, 2392, 2391, ...
## Resampling results across tuning parameters:
##
##    kmax  kernel       distance  Accuracy   Kappa
##     7    rectangular  1         0.5920162  0.10165397
##     7    rectangular  2         0.6127411  0.13113986
##     7    rectangular  3         0.5980425  0.10823747
##     7    cos          1         0.6085774  0.13398867
##     7    cos          2         0.6108330  0.13369349
##    7     cos          3           0.6014217  0.10781491
##    8     rectangular  1           0.6082242  0.11058317
##    8     rectangular  2           0.6089675  0.11770387
##    8     rectangular  3           0.6089166  0.12617523
##    8     cos          1           0.6104628  0.13582776
##    8     cos          2           0.6085717  0.12271845
##    8     cos          3           0.5987816  0.09818759
##    9     rectangular  1           0.6074723  0.10586840
##    9     rectangular  2           0.6089675  0.11770387
##    9     rectangular  3           0.6089166  0.12617523
##    9     cos          1           0.6142307  0.13032570
##    9     cos          2           0.6097208  0.11880103
##    9     cos          3           0.6051670  0.10552831
##   10     rectangular  1           0.6040888  0.09373252
##   10     rectangular  2           0.6134958  0.12311467
##   10     rectangular  3           0.6066651  0.10857344
##   10     cos          1           0.6138533  0.12365188
##   10     cos          2           0.6127255  0.12038380
##   10     cos          3           0.6063033  0.10432537
##   11     rectangular  1           0.6040888  0.09373252
##   11     rectangular  2           0.6134958  0.12311467
##   11     rectangular  3           0.6066651  0.10857344
##   11     cos          1           0.6168595  0.12468659
##   11     cos          2           0.6119722  0.11457817
##   11     cos          3           0.6115764  0.10952237
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were kmax = 11, distance = 1 and kerne
l
##  = cos.

#APPLYING THE MODEL
# Predict
pred_knn <- predict(kknn_fit, df_wp)
# Generate confusion matrix
cm <- confusionMatrix(df_wp$Potability, pred_knn)
cm

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   NonPotable Potable
##   NonPotable       1628      35
##   Potable           171     823
##
##                Accuracy : 0.9225
##                  95% CI : (0.9116, 0.9324)
##     No Information Rate : 0.6771
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.8298
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9049
##             Specificity : 0.9592
##          Pos Pred Value : 0.9790
##          Neg Pred Value : 0.8280
##              Prevalence : 0.6771
##          Detection Rate : 0.6127
##    Detection Prevalence : 0.6259
##       Balanced Accuracy : 0.9321
##
##        'Positive' Class : NonPotable
##

knn_results = kknn_fit$results

# group by k and distance function, create an aggregation by averaging
knn_results <- knn_results %>%
 group_by(kmax, kernel) %>%
 mutate(avgacc = mean(Accuracy))
head(knn_results)

## # A tibble: 6 × 8
## # Groups:   kmax, kernel [2]
##    kmax kernel      distance Accuracy Kappa AccuracySD KappaSD avgacc
##   <int> <fct>          <int>    <dbl> <dbl>      <dbl>   <dbl>  <dbl>
## 1     7 rectangular        1    0.592 0.102     0.0419  0.0900  0.601
## 2     7 cos                1    0.609 0.134     0.0158  0.0333  0.607
## 3     7 rectangular        2    0.613 0.131     0.0204  0.0422  0.601
## 4     7 cos                2    0.611 0.134     0.0161  0.0386  0.607
## 5     7 rectangular        3    0.598 0.108     0.0262  0.0572  0.601
## 6     7 cos                3    0.601 0.108     0.0306  0.0677  0.607

# plot aggregated (over Minkowski power) accuracy per k, split by distance fu
nction
ggplot(knn_results, aes(x=kmax, y=avgacc, color=kernel)) +
 geom_point(size=3) + geom_line()
```



```
#knn cluster
rotated_data$Color <- pred_knn
ggplot(data = rotated_data, aes(x=PC1, y=PC2, col = Color )) + geom_point(alp
ha = 0.3)
```

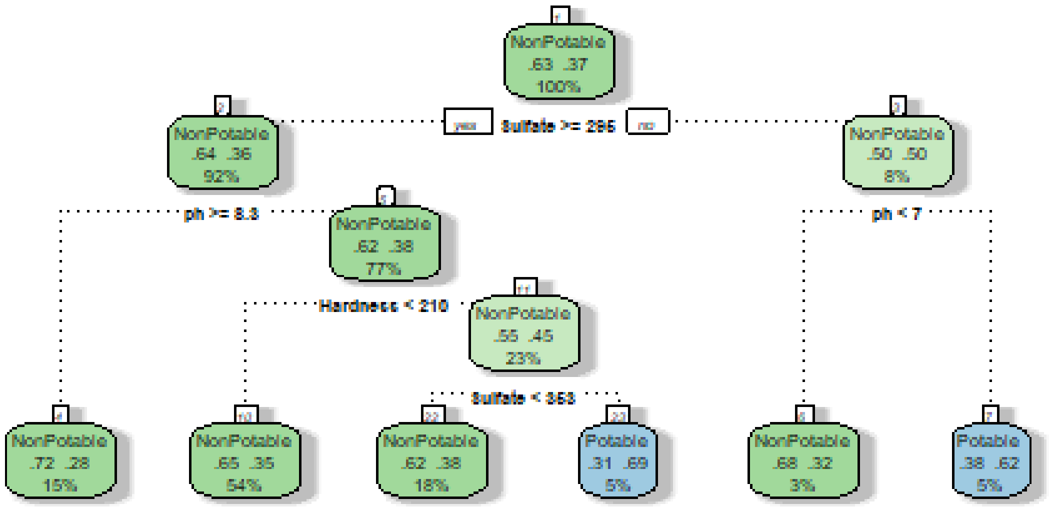
```
The confusion matrix of the knn tunegrid model performed the best. With bette
r accuracy in predicting.
```



2. DECISION TREE: Categorical variable decision tree. The decision tree technique, in contrast to other supervised learning methods, is capable of handling both classification and regression issues.

```
set.seed(123)
#Decision Tree

train_control = trainControl(method = "cv", number= 10) #evaluation method
tree_wp <- train(Potability ~., data = df_wp, method = "rpart",trControl = tr
ain_control) #fit model
tree_wp #Evaluate fit

## CART
##
## 2657 samples
##    9 predictor
##    2 classes: 'NonPotable', 'Potable'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2391, 2392, 2392, 2392, 2392, 2391, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.00804829  0.6337924  0.08629976
##   0.01106640  0.6383051  0.08236527
##   0.01670020  0.6273986  0.02520954
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0110664.

df_wp$Potability <- as.factor(df_wp$Potability)

#predict with test
pred_tree <- predict(tree_wp, df_wp)
#generate confusion matrix
confusionMatrix(df_wp$Potability, pred_tree)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   NonPotable Potable
##   NonPotable       1570      93
##   Potable           818     176
##
##                Accuracy : 0.6571
##                  95% CI : (0.6387, 0.6752)
##     No Information Rate : 0.8988
##     P-Value [Acc > NIR] : 1
##
```

```
##                  Kappa : 0.142
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6575
##             Specificity : 0.6543
##          Pos Pred Value : 0.9441
##          Neg Pred Value : 0.1771
##              Prevalence : 0.8988
##          Detection Rate : 0.5909
##    Detection Prevalence : 0.6259
##       Balanced Accuracy : 0.6559
##
##        'Positive' Class : NonPotable
##

#visualize tree
fancyRpartPlot(tree_wp$finalModel, caption = "")
```



THE DECISION TREE ABOVE RELIES ON THREE FEATURES ph, sulfates and hardness.

NOW LET US PERFORM MULTIPLE DECISION TREES BY VARYING PARAMETERS AND IDENTIFYING THE BEST TREE.

```r
set.seed(123)
# Partition the data
index = createDataPartition(y=df_wp$Potability, p=0.7, list=FALSE)
# Everything in the generated index list
train_set = df_wp[index,]
# Everything except the generated indices
test_set = df_wp[-index,]



#Initialize cross validation
train_control = trainControl(method = "cv", number = 10)
# Tree 1
hypers = rpart.control(minsplit = 2, maxdepth = 1, minbucket = 2)
tree1 <- train(Potability ~., data = train_set, control = hypers, trControl =
train_control, method = "rpart1SE")
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree1, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$Potability, pred_tree)
# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree1, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$Potability, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree1$finalModel$frame)
# Form the table
comp_tbl <- data.frame("Nodes" = nodes, "TrainAccuracy" = a_train, "TestAccur
acy" = a_test,
"MaxDepth" = 1, "Minsplit" = 2, "Minbucket" = 2)


# Tree 2
hypers = rpart.control(minsplit = 5, maxdepth = 2, minbucket = 5)
tree2 <- train(Potability ~., data = train_set, control = hypers, trControl =
train_control, method = "rpart1SE")
```

```r
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree2, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$Potability, pred_tree)
# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree2, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$Potability, pred_tree)
# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree2$finalModel$frame)
# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 2, 5, 5))


# Tree 3
hypers = rpart.control(minsplit = 50, maxdepth = 3, minbucket = 50)
tree3 <- train(Potability ~., data = train_set, control = hypers, trControl =
train_control, method = "rpart1SE")
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree3, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$Potability, pred_tree)
# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree3, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$Potability, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree3$finalModel$frame)
# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 3, 50, 50))


# Tree 4
hypers = rpart.control(minsplit = 100, maxdepth = 4, minbucket = 100)
tree4 <- train(Potability ~., data = train_set, control = hypers, trControl =
train_control, method = "rpart1SE")
```

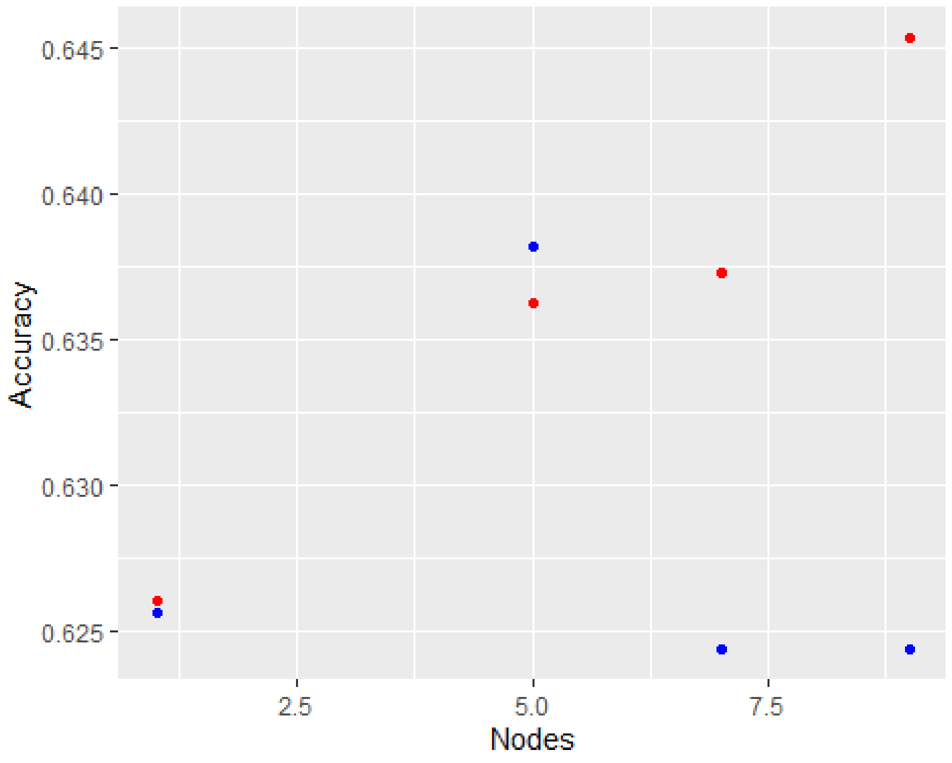
```r
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree4, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$Potability, pred_tree)
# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree4, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$Potability, pred_tree)
# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree4$finalModel$frame)
# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 4, 100, 100))


# Tree 5
hypers = rpart.control(minsplit = 500, maxdepth = 5, minbucket = 500)
tree5 <- train(Potability ~., data = train_set, control = hypers, trControl =
train_control, method = "rpart1SE")
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree5, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$Potability, pred_tree)
# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree5, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$Potability, pred_tree)
# Get training accuracy

a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree5$finalModel$frame)
# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 5, 500, 500))


# Tree 6
hypers = rpart.control(minsplit = 1000, maxdepth = 6, minbucket = 1000)
tree6 <- train(Potability ~., data = train_set, control = hypers, trControl =
train_control, method = "rpart1SE")
```

```r
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree6, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$Potability, pred_tree)
# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree6, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$Potability, pred_tree)
# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree6$finalModel$frame)
# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 6, 1000, 1000))


# Tree 7
hypers = rpart.control(minsplit = 2000, maxdepth = 7, minbucket = 2000)
tree7 <- train(Potability ~., data = train_set, control = hypers, trControl =
train_control, method = "rpart1SE")
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree7, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$Potability, pred_tree)
# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree7, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$Potability, pred_tree)
# Get training accuracy
a_train <- cfm_train$overall[1]

# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree7$finalModel$frame)
# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 7, 2000, 2000))


# Tree 8
hypers = rpart.control(minsplit = 5000, maxdepth = 10, minbucket = 5000)
tree8 <- train(Potability ~., data = train_set, control = hypers, trControl =
train_control, method = "rpart1SE")
```

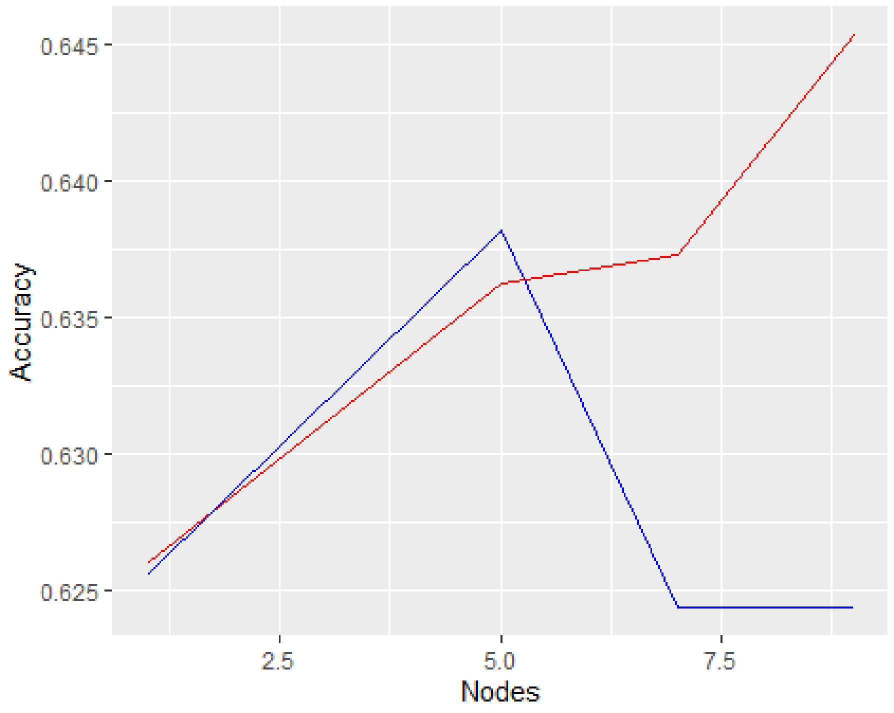
```r
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree8, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$Potability, pred_tree)
# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree8, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$Potability, pred_tree)
# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree8$finalModel$frame)
# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 10, 5000, 5000))


# Tree 9
hypers = rpart.control(minsplit = 10000, maxdepth = 20, minbucket = 10000)
tree9 <- train(Potability ~., data = train_set, control = hypers, trControl =
train_control, method = "rpart1SE")
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree9, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$Potability, pred_tree)
# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree9, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$Potability, pred_tree)
# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy

a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree9$finalModel$frame)
# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 20, 10000, 10000)
)


# Tree 10
hypers = rpart.control(minsplit = 15000, maxdepth = 30, minbucket = 15000)
tree10 <- train(Potability ~., data = train_set, control = hypers, trControl
= train_control, method = "rpart1SE")
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree10, train_set)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set$Potability, pred_tree)
# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree10, test_set)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set$Potability, pred_tree)
# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree10$finalModel$frame)
# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 50, 15000, 15000)
)
comp_tbl
```

```
##          Nodes TrainAccuracy TestAccuracy MaxDepth Minsplit Minbucket
## Accuracy     1     0.6260075    0.6256281        1        2         2
## 1            5     0.6362171    0.6381910        2        5         5
## 11           9     0.6453520    0.6243719        3       50        50
## 12           7     0.6372918    0.6243719        4      100       100
## 13           1     0.6260075    0.6256281        5      500       500
## 14           1     0.6260075    0.6256281        6     1000      1000
## 15           1     0.6260075    0.6256281        7     2000      2000
## 16           1     0.6260075    0.6256281       10     5000      5000
## 17           1     0.6260075    0.6256281       20    10000     10000
## 18           1     0.6260075    0.6256281       50    15000     15000
```

```r
# Visualize with scatter plot
ggplot(comp_tbl, aes(x=Nodes)) +
geom_point(aes(y = TrainAccuracy), color = "red") +
geom_point(aes(y = TestAccuracy), color="blue") +
ylab("Accuracy")
```



```r
# Visualize with line plot
ggplot(comp_tbl, aes(x=Nodes)) +
geom_line(aes(y = TrainAccuracy), color = "red") +
geom_line(aes(y = TestAccuracy), color="blue") +
ylab("Accuracy")
```



THE TREE WITH 5 NODES PRODUCED THE BEST ACCURACY AND KAPPA for both test and train. ALSO, A TREE THAT CAN BE EASILY INFERRED COMPARED TO THE FIRST ONE.

Although there is no comparative difference in the confusion matrix between the first model and the current chosen one.

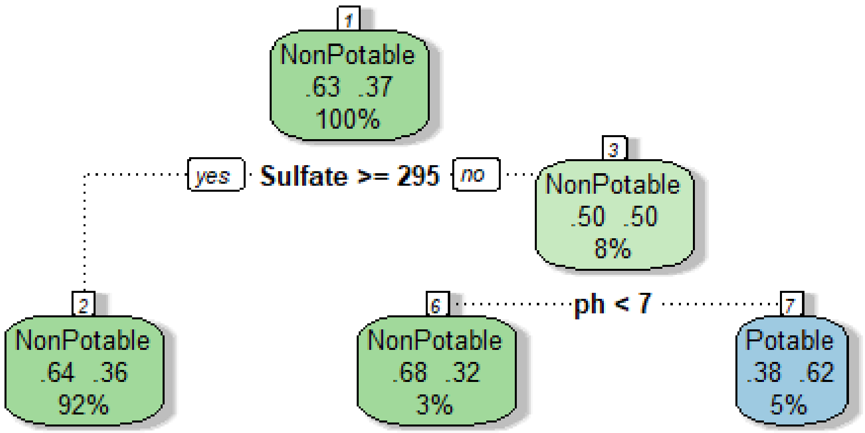
```
train_control = trainControl(method = "cv", number= 10)
# Tree 2 Final Model
hypers = rpart.control(minsplit = 5, maxdepth = 2, minbucket = 5)
tree2 <- train(Potability ~., data = df_wp, control = hypers, trControl = tra
in_control, method = "rpart1SE")
tree2
```

```
## CART
##
## 2657 samples
##    9 predictor
##    2 classes: 'NonPotable', 'Potable'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2391, 2391, 2392, 2391, 2391, 2391, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6311891  0.04990252
```

```
# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree2, df_wp)
# Confusion Matrix
cfm <- confusionMatrix(df_wp$Potability, pred_tree)
cfm
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction   NonPotable Potable
##   NonPotable       1612      51
##   Potable           911      83
##
##                Accuracy : 0.6379
##                  95% CI : (0.6193, 0.6562)
##     No Information Rate : 0.9496
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.064
##
##  Mcnemar's Test P-Value : <2e-16
##
```

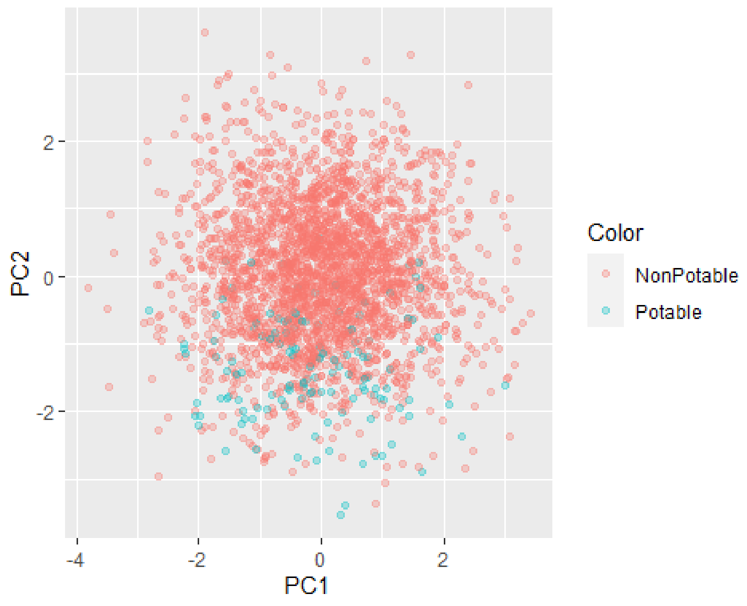
```
##             Sensitivity : 0.6389
##             Specificity : 0.6194
##          Pos Pred Value : 0.9693
##          Neg Pred Value : 0.0835
##              Prevalence : 0.9496
##          Detection Rate : 0.6067
##    Detection Prevalence : 0.6259
##       Balanced Accuracy : 0.6292
##
##        'Positive' Class : NonPotable
##

#visualize tree
fancyRpartPlot(tree2$finalModel, caption = "")
```



THE DECISION TREE ABOVE RELIES ON TWO MAIN FEATURES – ph and Sulfate.

```
# Remove class labels
predictors <- df_wp %>% select(-c(Potability))
head(predictors)

##          ph Hardness   Solids Chloramines  Sulfate Conductivity Organic_ca
rbon
## 1  7.036752 204.8905 20791.32    7.300212 368.5164     564.3087       10.37
9783
## 3  8.099124 224.2363 19909.54    9.275884 333.0735     418.6062       16.86
8637
```

```
## 4  8.316766 214.3734 22018.42    8.059332 356.8861      363.2665       18.43
6524
## 5  9.092223 181.1015 17978.99    6.546600 310.1357      398.4108       11.55
8279
## 6  5.584087 188.3133 28748.69    7.544869 326.6784      280.4679        8.39
9735
## 7 10.223862 248.0717 28749.72    7.513408 393.6634      283.6516       13.78
9695
##   Trihalomethanes Turbidity
## 1        86.99097  2.963135
## 3        66.42009  3.055934
## 4       100.34167  4.628771
## 5        31.99799  4.075075
## 6        54.91786  2.559708
## 7        84.60356  2.672989

# Center scale allows us to standardize the data
preproc <- preProcess(predictors, method=c("center", "scale"))
# We have to call predict to fit our data based on preprocessing
predictors <- predict(preproc, predictors)
# Calculate PCA
pca = prcomp(predictors)
# Save as dataframe
rotated_data = as.data.frame(pca$x)
# Add original labels as a reference
rotated_data$Color <- df_wp$Potability

#decision tree
rotated_data$Color <- pred_tree
ggplot(data = rotated_data, aes(x=PC1, y=PC2, col = Color )) + geom_point(alp
ha = 0.3)
```

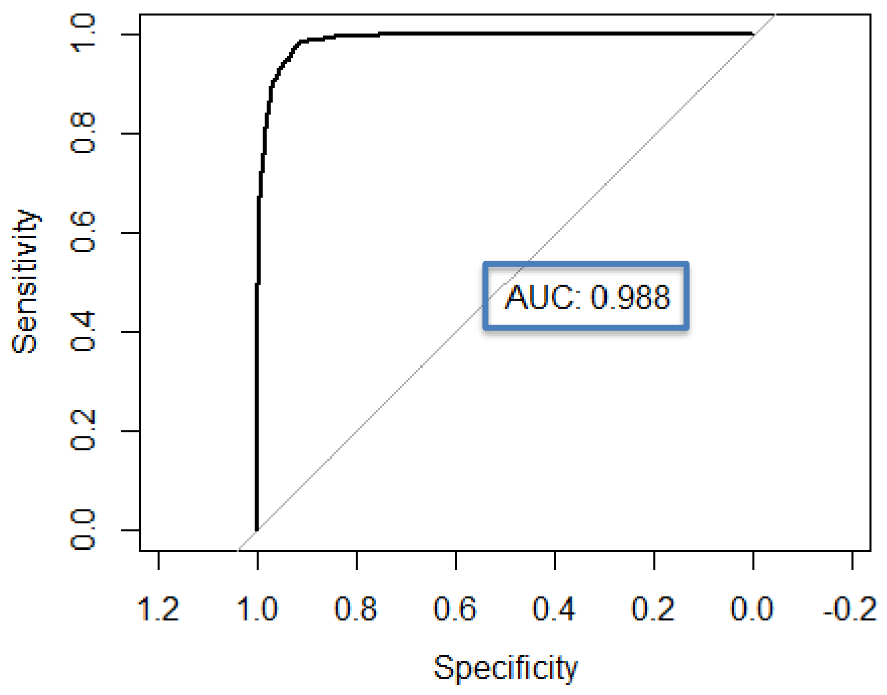


G. <u>EVALUATION</u>

Knn METRICS

```
cm

## Confusion Matrix and Statistics
##
##                Reference
## Prediction   NonPotable Potable
##   NonPotable       1628      35
##   Potable           171     823
##
##                Accuracy : 0.9225
##                  95% CI : (0.9116, 0.9324)
##     No Information Rate : 0.6771
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8298
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9049
##             Specificity : 0.9592
##          Pos Pred Value : 0.9790
##          Neg Pred Value : 0.8280
##              Prevalence : 0.6771
##          Detection Rate : 0.6127
##    Detection Prevalence : 0.6259
##       Balanced Accuracy : 0.9321
##
##        'Positive' Class : NonPotable
##

# Store the byClass object of confusion matrix as a dataframe
metrics <- as.data.frame(cm$byClass)
# View the object
metrics

##                       cm$byClass
## Sensitivity            0.9049472
## Specificity            0.9592075
## Pos Pred Value         0.9789537
## Neg Pred Value         0.8279678
## Precision              0.9789537
## Recall                 0.9049472
## F1                     0.9404968
## Prevalence             0.6770794
## Detection Rate         0.6127211
## Detection Prevalence   0.6258939
## Balanced Accuracy      0.9320773
```

```
pred_prob <- predict(kknn_fit, df_wp, type = "prob")
roc_obj <- roc((df_wp$Potability), pred_prob[,1])

## Setting levels: control = NonPotable, case = Potable

## Setting direction: controls > cases

plot(roc_obj, print.auc=TRUE)
```



DECISION TREE METRICS

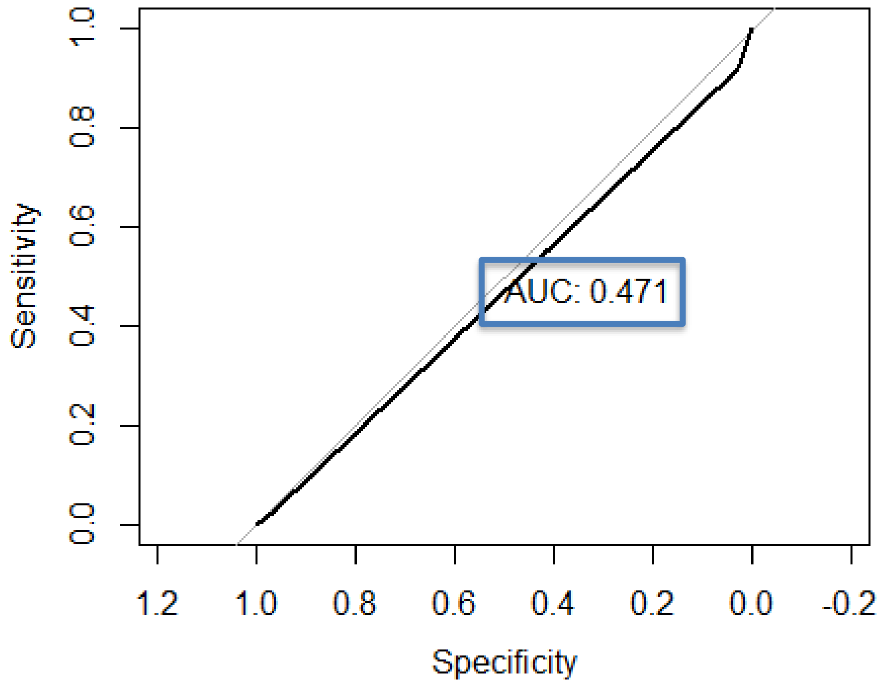
```
# Confusion Matrix
cfm <- confusionMatrix(df_wp$Potability, pred_tree)
cfm

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   NonPotable Potable
##   NonPotable       1612      51
##   Potable           911      83
##
##                Accuracy : 0.6379
##                  95% CI : (0.6193, 0.6562)
##     No Information Rate : 0.9496
##     P-Value [Acc > NIR] : 1
```

```
##
##                Kappa : 0.064
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6389
##             Specificity : 0.6194
##          Pos Pred Value : 0.9693
##          Neg Pred Value : 0.0835
##              Prevalence : 0.9496
##          Detection Rate : 0.6067
##    Detection Prevalence : 0.6259
##       Balanced Accuracy : 0.6292
##
##        'Positive' Class : NonPotable
##
```

```
metrics <- as.data.frame(cfm_train$byClass)
# View the object
metrics
```

```
##                      cfm_train$byClass
## Sensitivity                 0.63892192
## Specificity                 0.61940299
## Pos Pred Value              0.96933253
## Neg Pred Value              0.08350101
## Precision                   0.96933253
## Recall                      0.63892192
## F1                          0.77018634
## Prevalence                  0.94956718
## Detection Rate              0.60669928
## Detection Prevalence        0.62589387
## Balanced Accuracy           0.62916245
```

```
pred_prob <- predict(tree2, df_wp, type = "prob")
roc_obj <- roc((df_wp$Potability), pred_prob[,1])

## Setting levels: control = NonPotable, case = Potable

## Setting direction: controls < cases

plot(roc_obj, print.auc=TRUE)
```



EVALUATION OBSERVATIONS:

As we can see that the knn model performs far better than decision tree, the confusion matrix of knn provides a better accuracy and kappa,

the ROC curve plot for knn - almost nearing one at the y-axis proving that it is the better model with an AUC of 0.988. It also has better sensitivity and specificity compared to decision tree metrics.

H. REPORT ANALYSIS:

1. The models could have performed better if the data present in the Features were more consistent and accurate in distinguishing the two classes of the target variable. (Potability)

2. The features consisted of discrepancies, as the data in certain features were exceeding the standards for class Potable.

3. The features had – more than 90% of the data was considered Hard. Less than 5% was safe in terms of chloramine and sulfate levels. And more than 90% of water sample had higher carbon levels. 90% was safe in terms of turbidity and more than 75% was safe in terms of Trihalomethane levels.

4. The correlation coefficients between the features very low.

5. Knn with tune grid performed the best in terms of prediction compared to decision tree. Providing an accuracy of 0.9225. The training accuracy is 0.6168595. Decision tree performs slightly better in terms of training accuracy but fails while prediction and the confusion matrix provide insights of a non-reliable model. We can take a look at this in terms of ROC plots as well.

6. Knn has a Precision of 0.9789537 and a Recall of 0.9049472.

7. Decision tree has a Precision of 0.96933253 and a Recall of 0.63892192

Key Takeaway: The data should be more precise and accurately reported to perform and build models that can provide accurate readings.

I.REFLECTION:

The amount of data has vastly increased and is ever growing. Data complexity is increasing over time in a similar manner. Today, a data scientist works with multiple data formats simultaneously to predict and draw conclusions. Due to the complexity there is currently a desire for techniques, procedures, or tools that can help them evaluate data more quickly and easily.

Data Science is the integration of sophisticated Machine Learning algorithms with a wide range of tools to assist data scientists in making decisions, seeing novel trends, and developing novel approaches to predictive analysis.

Machine learning is based on the principle that you can educate and train machines by giving them data and specifying features. When given fresh, pertinent data, computers learn, grow, adapt, and develop on their own without the need for explicit programming. Machine learning is a relatively limited field without data. The Machine observes the dataset, spots patterns in it, automatically picks up on patterns from behavior, and predicts outcomes. Few limitations to overcome in machine learning are lack of training data, model scalability and data discrepancies.

From the report above the main key components such as regression analysis, clustering and classification bring us closer to building models that can benefit mankind in numerous fields by reducing workload and saving time.